DTIC FILE COPY

1

AD-A202 710

ROBOTIC COMPLIANT MOTION CONTROL FOR AIRCRAFT REFUELING APPLICATIONS

THESIS

David J. Duvall
Captain, USAF

AFIT/GA/ENG/88D-1

DTIC
ELECTE
JAN 1 8 1989
S D
cD

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

89 1 17 092

①

AFIT/GA/ENG/88D-1

DTIC
S ELECTE D
JAN 1 8 1989
D

ROBOTIC COMPLIANT MOTION CONTROL FOR AIRCRAFT REFUELING APPLICATIONS

THESIS

David J. Duvall
Captain, USAF

AFIT/GA/ENG/88D-1

Approved for public release; distribution unlimited.

AFIT/GA/ENG/88D-1

ROBOTIC COMPLIANT MOTION CONTROL FOR AIRCRAFT REFUELING APPLICATIONS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Astronautical Engineering

David J. Duvall, B.S.
Captain, USAF

December, 1988

DTIC
COPY
INSPECTED
6

Accesion For
NTIS CRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution
Availability Codes
Dist | Avail and/or Special
A-1

Approved for public release; distribution unlimited.

Preface

A principal objective of the AFIT robotics research program is the development of compliant motion techniques for Air Force applications. Compliant motion is an essential capability for any robot seeking to emulate a human, since controlled compliance is needed in any task requiring significant interaction between the manipulator (either human or robot) and the environment. Refueling of aircraft with a robot manipulator is one application requiring compliant motion for successful completion. In the AFIT robotics program, as well as this thesis, the refueling problem serves as a "strawman" task for orienting compliant motion research.

The goal of this thesis was to create a hardware and software environment allowing for the development and testing of compliant motion control techniques on an industrial manipulator. With this environment in hand further research into the actual compliant motion control algorithms can proceed. In this thesis a preliminary version of an impedance control law was used to demonstrate the environment, and to perform an initial investigation into compliant motion control of a PUMA 560 robot arm. This initial impedance control implementation has a number of features which can, and should be, improved.

My thesis would not have been possible without the sponsorship of Dr. Mangal Chawla of the AFWAL Flight Dynamics Laboratory. I would like to thank Dr. Chawla and his staff for the technical information on the proposed robot refueler, and for supplying the force sensor electronics. I am similarly indebted to Mr. Dexter Kalt of the ASD Fuel and Hazards branch for his information on aerial refueling systems.

Closer to home, I would like to say thanks to Captain Larry Tellman and Mr. Don Smith for teaching me enough to survive as my own electronics technician in the lab. Captain Steve Parker deserves thanks for showing me how to use MACSYMA and LATEX, but more importantly I valued his friendship and encouragement throughout my entire AFIT experience.

The *key man* in making this all happen was my advisor, Dr./Captain Mike Leahy. Without his teaching, ideas, and support I could never have completed the first assembly code subroutine, let alone this entire project. Of course, the *key woman* in helping me through has been my wife, [redacted] I am continually amazed at how she found the energy to get her own degree, work nearly full time, and still look after me — it must have been love.

David J. Duvall

Table of Contents

Page

Preface ii

Table of Contents iv

List of Figures viii

List of Tables x

Abstract xi

I. Introduction 1-1
  1.1 Motivation 1-1
  1.2 Objective 1-2
  1.3 Problem Statement 1-2
    1.3.1 Background 1-2
    1.3.2 Thesis Research Goals 1-5
  1.4 Method of Approach 1-6
  1.5 Contribution and Summary 1-8
  1.6 Organization 1-10

II. Literature Review and Control Law Selection 2-1
  2.1 Force Sensor Design and Calibration 2-1
  2.2 Compliant Motion Control Laws 2-3
    2.2.1 Review of Stiffness Control 2-3
    2.2.2 Review of Hybrid Control 2-8
    2.2.3 Review of Impedance Control 2-15

Page

2.3 Additional Compliant Motion Topics 2-23

2.3.1 Effects of Friction 2-23

2.3.2 Additional Stability Issues 2-25

2.4 Control Law Selection 2-30

III. Algorithm Development 3-1

3.1 Impedance Control 3-1

3.1.1 The Concepts of Impedance and Admittance 3-1

3.1.2 Desired Dynamics 3-6

3.1.3 Manipulator Dynamics 3-10

3.1.4 Control Law 3-11

3.1.5 Simplified Control Law 3-12

3.1.6 Development of Control Law Components . . 3-14

3.2 Force Sensing 3-20

3.2.1 Force Data Scaling 3-20

3.2.2 Calibration and External Force in the Sensor Frame 3-21

3.2.3 Limit Checking 3-22

3.2.4 Transformation to Tool Frame 3-22

3.2.5 Transformation to World Coordinates and Gravity Correction 3-25

3.2.6 Determining Calibration Constants 3-29

3.2.7 Determining Sensed Mass 3-31

3.3 Control of Joints 1, 4, 5, and 6 3-33

3.4 Summary 3-34

IV. Implementation 4-1

4.1 PUMA 560 4-2

4.2 Force Sensor 4-3

Page

4.3 Refueling Port and Nozzle Mockup 4-11

4.4 Hierarchial Control System 4-17

4.4.1 Organizer Level 4-19

4.4.2 Coordinator Level 4-23

4.5 Summary 4-32

V. Evaluation 5-1

5.1 Force Sensing Performance 5-1

5.1.1 Force Sensor Resolution 5-2

5.1.2 Force Sensing Accuracy 5-2

5.1.3 Force Sensing Error Sources 5-3

5.2 Timing 5-4

5.2.1 Timing Tests 5-5

5.2.2 Sample Time Effects 5-7

5.3 Trajectory Tracking 5-9

5.3.1 Results 5-10

5.3.2 Discussion of Trajectory Tracking Results . . 5-21

5.4 Compliant Motion Tests 5-27

5.4.1 Compliant Motion Test Using Active Compliance 5-28

5.4.2 Compliant Motion Test Using Passive Compliance 5-33

5.5 Discussion of Evaluation Results 5-38

5.6 Summary 5-41

VI. Conclusions and Recommendations 6-1

6.1 Conclusions 6-1

6.2 Recommendations 6-4

Page

A. Control Law Derivation A-1

B. Aircraft Robotic Refueling Background B-1

  B.1 Concept Overview B-1

  B.2 Aerial Refueling Port and Nozzle Description B-5

C. New and Modified Software C-1

  C.1 Organizer Software C-1

  C.2 Coordinator Software for the Servo Processor C-10

  C.3 Coordinator Software for the Parallel Processor C-25

Bibliography BIB-1

Vita VITA-1

List of Figures

Figure **Page**

2.1. Scheinman Wrist Force Sensor 2-2

2.2. Stiffness Control System 2-7

2.3. Natural and Artificial Constraints for Turning a Screwdriver . . 2-9

2.4. Hybrid Control Architecture 2-11

2.5. Hogan's Virtual Trajectory and Ideal Force Response 2-20

2.6. Hogan's Impedance Controller Performance 2-21

2.7. Inner and Outer Control Loops 2-29

3.1. Electrical Network 3-2

3.2. Mechanical Spring of Stiffness K, at Position X, Applying Force F 3-3

3.3. Mass M, Acted on by Force F, Being Displaced to Position X . . 3-3

3.4. Target Dynamics Model 3-7

3.5. True Moments vs. Sensed Moments 3-24

3.6. Tool and Sensor Diagram 3-26

4.1. PUMA 500 Series Robot Arm with Digital Controller and Teach Pendant 4-4

4.2. Link Coordinates and Denavit-Hartenberg Parameters for the PUMA 560 4-5

4.3. Force Sensor System 4-6

4.4. Force Transducer 4-7

4.5. Force Sensing Element 4-8

4.6. Force Sensor Power Supply Network 4-12

4.7. Refueling Port Mockup and Platform 4-14

4.8. Refueling Port Mockup, Detailed View 4-15

Figure Page

4.9. Nozzle Mockup 4-16
4.10. Centers of Gravity and Tool Lengths 4-18
4.11. Hierarchical Control System Hardware 4-20

5.1. Desired Linear Trajectory 5-11
5.2. Desired Circular Trajectory 5-12
5.3. Stationary Trajectory Tracking Results 5-14
5.4. Typical Tool Position Tracking for the Linear Trajectory. 5-16
5.5. Position and Position Error vs. Time, Linear Trajectory 5-17
5.6. Typical Tool Position Tracking for the Circular Trajectory . . . 5-19
5.7. Position and Position Error vs. Time, Circular Trajectory 5-20
5.8. Velocity Effect on the X and Z Position Error 5-23
5.9. X and Z Position Error with Four Interpolations per Set Point . 5-25
5.10. Tool Path for the Constrained Circular Trajectory Using Active Compliance 5-30
5.11. Tool Position with Respect to Time for the Constrained Circular Trajectory Using Active Compliance 5-31
5.12. Interface Force for the Constrained Circular Trajectory Using Active Compliance 5-32
5.13. Tool Path for the Constrained Circular Trajectory Using Passive Compliance 5-35
5.14. Tool Position with Respect to Time for the Constrained Circular Trajectory Using Passive Compliance 5-36
5.15. Interface Force for the Constrained Circular Trajectory Using Passive Compliance 5-37

B.1. Robot refueling F-16 through aerial refueling port B-3
B.2. Outer End of KC-10A Refueling Boom Including Nozzle B-6
B.3. Aerial Refueling Nozzle B-7
B.4. Aerial Refueling Receptacle B-9

List of Tables

Table Page

3.1. J_{min} Values for the PUMA 560 Wrist 3-34

4.1. JR3 Sensor Force and Moment Limits 4-9

4.2. Tool and Payload Parameters 4-17

4.3. Organizer Subroutines 4-22

4.4. Coordinator Subroutines 4-26

4.5. Parallel Processor Subroutines 4-32

5.1. Sensor Scale Values and Corresponding Resolution 5-2

5.2. F_{int} Error Data from the Force Sensor Accuracy Tests 5-3

5.3. Control Algorithm Execution Times 5-5

5.4. Stationary Trajectory Test, Controller Parameters 5-13

5.5. Linear Trajectory Test, Controller Parameters 5-15

5.6. Circular Trajectory Test, Controller Parameters 5-18

5.7. Controller Parameters for Velocity Effects Tests. 5-22

AFIT/GA/ENG/88D-1

Abstract

A promising Air Force application of robotics technology is the refueling of aircraft or spacecraft with a robotic manipulator. The refueling task is fundamentally a component assembly task, and like many other assembly tasks, successful completion requires compliance between the manipulator and the environment for success. Compliant motion control techniques, such as impedance control, use force feedback to generate active compliance in the robot manipulator. In this thesis a compliant motion control environment was established, and a simplified, preliminary version of an impedance control law was implemented. The compliant motion environment employs three digital processors in a hierarchial control structure to command a PUMA 560 robot arm. Applied force and moment information are provided by a wrist mounted, three axis force sensor. An original method was developed to transform forces and moments acting on the tool, measured in the sensor frame, to the cartesian world coordinate frame. This method eliminates the forces and moments caused by the tool weight from the measured values. The concept of impedance is explained, and motivated as the basis for compliant motion control. The theoretical development leading to the simplified impedance control law is presented. The simplified impedance control law was used to provide active compliance for links 2 and 3 of the PUMA arm. The remaining four links of the PUMA were not actively used in the impedance control experiment. Force sensor accuracy was experimentally quantified and found to be sensitive to errors in arm calibration. Execution times were determined for the major components of the impedance control algorithm. Initial testing demonstrated the ability of the impedance controller to use active compliance to reduce interface forces. Elementary compliant motion was achieved, however trajectory tracking performance requires improvements. Problem areas causing poor tracking performance are identified, and possible solutions are recommended.

ROBOTIC COMPLIANT MOTION CONTROL FOR AIRCRAFT REFUELING APPLICATIONS

I. Introduction

1.1 Motivation

One promising application of robotics technology is the refueling of aircraft or spacecraft with a robot manipulator. The Air Force Flight Dynamics Laboratory has recently completed a preliminary conceptual study of the possible benefits of aircraft robotic refueling [3]. Additionally, NASA and the Air Force have considered the use of tele-robots or autonomous robots for extending the life of spacecraft through on-orbit refueling [45]. These studies show the potential robotic systems have for saving money, manpower, and lives.

Robot refueling is fundamentally a component assembly task requiring the robot to mate a fuel line nozzle with a refueling port. Current industrial robots rely primarily on position feedback controllers. However once the robot refueler nozzle contacts the aircraft, position feedback control will not be adequate because large contact forces may develop, possibly leading to robot controller instability or physical damage. Compliant motion control solves these problems by considering force feedback in the control law of an intelligent robot. Clearly a compliant motion capability is necessary for performing the robot refueling task or any other precision assembly task.

1.2 Objective

Further broad-based research into intelligent robotics capable of emulating human arm motion requires the development of a compliant motion capability. The objective of this study is to create a bread-board compliant motion environment and implement an initial compliant motion controller. The results will lay a foundation of knowledge and equipment for further research leading to a demonstration of robotic aircraft refueling.

1.3 Problem Statement

1.3.1 Background.

1.3.1.1 What is Compliant Motion Control. Based on the research of Goertz , Paul identifies three states of manipulator motion for any assembly problem,

"Motion in free space" when the manipulator motion is unconstrained by contact with the environment

"Contact" when the manipulator switches modes between unconstrained and constrained motion

"Exertion of a force" when the manipulator motion is constrained by contact with the environment [44, p. 1967]

The robot refueler's task is no different and also consists of these three phases.

- Phase I – the robot uses a vision system to search for port and guide the nozzle to a point close to the slipway or port.
- Phase II – the nozzle makes initial contact with the slipway and the robot begins to slide the nozzle along the slipway while impact transients damp out.

- **Phase III** – the robot continues to slide the nozzle along the slipway and then inserts the nozzle in the port.

According to Kazerooni, "In constrained maneuvers, the manipulator is driven in its workspace so the environment continuously exerts a dynamic or kinematic constraint on the manipulator motion [26, p. 83]." Compliant motion control is the technique used to drive the manipulator so as to accomodate the constraint while accomplishing the task objectives. With compliant motion control, both the manipulator's position and *the constraint forces* are used to determine the arm control inputs[1]. The use of constraint force information to actively control the arm uniquely separates compliant motion control from other forms of robot control.

1.3.1.2 Why Use Compliant Motion Control. Current industrial arms typically use only position feedback to control the arm. Position feedback is appropriate for transport type tasks where objects are moved from one position to another, with little or no environmental contact (e.g. bin picking and spray painting). However, complex assembly tasks must be highly structured if they are performed with a position controlled arm. A highly structured task demands parts always be presented to the robot in the same position and orientation, possibly through the use of special assembly jigs or tight tolerances on part manufacturing and placement. Unfortunately, tasks can never be perfectly structured all the time, and when the structure breaks down some form of sensing is needed for the robot to determine what to do. Paul states force sensing is fundamental since it can provide vital information indicating when something has gone wrong (occurence of an unexpected force), and alternately when things are functioning correctly (no unexpected forces)[44, p. 1966]. An excellent example of a human using force feedback is a carpenter driving a nail through a wall and into an unseen beam. The carpenter expects a relatively uniform, moderate resistive force, indicating the nail has remained within the beam. A sudden, unexpected drop in resistance indicates the nail has missed the beam. A sudden rise in resistive force may indicate the nail head is flush with the wall or it may indicate the point has hit an obstacle. In this last case, the carpenter may use position feedback (how far in is the nail?) to determine which of the two events has occured.

---

[1] Throughout this document the term *force* is meant to include both forces and moments; the term *position* refers to both position and orientation. Exceptions to this rule should be apparent from the context of their use.

Most importantly, compliant motion control is needed to insure stability when constraint forces are imposed on the manipulator. Kazerooni highlights a major problem with position control when he says, "...the compensator treats the interaction forces as disturbances and tries to reject them, thus causing more interaction forces and torques. Saturation (of actuators), instability, and physical failure are the consequences...[26, p. 84]." On the other hand, a properly designed compliant motion controller is designed for stability in the presence of these constraint forces, and can actually use them to advantage in guiding the robot to complete a task.

1.3.1.3 *Compliant Motion Control Techniques.* Whitney divides compliant motion control techniques into two categories, *passive* control and *active* control [58, p. 5]. Passive control employs mechanical devices to change the compliance between the end-effector and the environment. One such device for doing this is the *Remote Center of Compliance* (RCC) [15]. The RCC attaches to the end-effector and acts like a six-dimensional spring with a pre-determined set of stiffnesses resisting translation and rotation about three axes. Passive compliance devices suffer from an inability to be reprogrammed with different stiffness values, resulting in a need to change devices when consecutive tasks require different amounts of compliance. Also, if the stiffness is low, then the uncertainty in knowledge of the tool or end-effector position is large because the spring is easily moved, and as a result position can not be accurately commanded [44, pp. 1969-1970].

According to Whitney, *active* control uses information about the current state of the task (position, velocity, applied constraint force) in a feedback control loop to calculate actuator commands [58, p. 6]. These commands drive the manipulator links to satisfy a set of desired position, velocity, and/or force interactions with the environment, resulting in compliant motion of the robot arm. Research has focused on active control techniques because they allow the user to more completely specify the state of the task, and changes in compliance can be rapidly made through software logic. Active force control techniques have the disadvantages of being more complex and computationally difficult. Numerous active control techniques have been proposed during the last 15 years [58, p. 6].

1.3.2 *Thesis Research Goals.* Research on compliant motion issues at AFIT is presently prevented by the lack of a suitable compliant motion control environment. Currently AFIT's PUMA 560 robot is run by the ARCADE[2] hierarchial control environment. ARCADE is a derivative of the RHCS/R3AGE[3] developed at the Rensselaer Polytechnic Institute [39]. ARCADE and the RHCS/R3AGE were designed for research into model based robot control and are well suited for that type of research[4]. However they can not support compliant motion because they do not have a force sensing capability.

A major goal of this thesis is to create a compliant motion environment by adding hardware and software for force sensing to the existing ARCADE environment. Techniques must also be created for transforming the raw force sensor data into an accurate picture of the constraint forces between the robot and environment. With the augmented environment, compliant motion control laws can be implemented on the PUMA 560 arm. A second goal of this thesis is to perform a preliminary investigation of compliant motion control using a simplified control law on links 2 and 3 of the PUMA arm.

---

[2] ARCADE is the AFIT Robotic Control Algorithm Development and Evaluation Environment

[3] RHCS/R3AGE is the Robotics and Automation Laboratory (RAL) Hierarchial Control System / RAL Real-time Robotic Algorithm Exerciser

[4] Model based robot control uses position feedback control loops, augmented with feedforward inputs calculated from a model of the robot dynamics.

1.4 Method of Approach.

The twin goals of creating a compliant motion control environment and a preliminary implementation of a compliant motion control law were approached in a parallel fashion. The preliminary control law served as a "strawman" control law for purposes of designing the control environment.

Before selecting a control law or defining a control environment, a literature review was performed to develop a knowledge base of compliant motion control techniques. Standard force sensing techniques were also reviewed since force sensor hardware needed to be incorporated into the environment. Lastly, a review of current aircraft aerial refueling equipment and a proposed robotic refueling concept provided background information for orienting the design of experimental hardware toward a demonstration of robotic aircraft refueling.

After completion of the literature review an appropriate control law was chosen for implementation in the control environment. A simplified, two degree of freedom version of an existing control law was desired because of the limited time available for this study. Use of the two degree of freedom control law allowed testing of the basic premise for the control law, although the resulting two degree of freedom controller remained scalable to greater degrees of freedom. By simplifing the control law its computational complexity was reduced, enabling it to be implemented with existing computer resources. The component matrices and vectors of the control law equation were computed from equations symbolically reduced using the MACSYMA symbolic manipulation, computer aided design (CAD) package [51]. Software for evaluating the control law needed to be designed to minimize computation time in order to allow high control loop sample rates and insure good digital controller performance. The control law has been implemented so as to allow for future expansion to a more complex, six degree of freedom control law.

Having picked a control law, it was possible to design a control environment using the support requirements of the chosen control law as examples of the support the environment should provide. Patterning the new compliant motion environment after the existing control environment, ARCADE, resulted in a hierarchially structured compliant motion environment. The hierarchial structure most effectively used the different capabilities of each computer in the control system. Modular software design facilitated future use and modification of the environment, while the use of existing software wherever possible preserved commonality between the compliant and existing environments.

The first principal task in developing the new environment was integrating a force sensor with the existing equipment, and developing the software to transform force sensor data into the force information required by the control law. Because of schedule constraints, the choice of an off-the-shelf vendor supplied force sensor was mandatory. After procuring the sensor it was integrated with the existing hardware by establishing an appropriate network of power supplies and communication links. New software to allow parallel communication with the digital controller was also created. An original theoretical method for scaling, calibrating, and transforming the "raw" sensor data into the desired vector of constraint forces was developed based on analysis of the manipulator kinematics and the sensor output. The force sensor did not provide a satisfactory method for separating the gravitational force on the tool from the measurements of constraint force, so this problem was corrected by devising a scheme to compensate the sensor data for the tool weight. These theoretical procedures were coded into a set of software routines for calibrating the sensor prior to run time, and also for using the sensor data in the force feedback portion of a compliant motion control law.

The second principal task in developing the compliant motion control environment involved creating a physical environment for testing compliant motion control techniques. To do this a tool and surface for it to interact with were designed and fabricated. The tool is modular to allow it to be reconfigured for different types of tests. Also, the test surface has a geometry supporting contact, slide, and insertion tests of varying difficulty. Since a long-range objective is the demonstration of robotic aircraft refueling, the tool and test surface were designed to model existing aircraft refueling hardware.

After the modular software elements of the control environment were developed they needed to be integrated with the control law software to form a complete compliant motion controller capable of commanding the PUMA arm. Testing of the compliant motion controller quantified controller performance. Limited testing is performed since the chosen control law was primarily implemented as an example, and only in a preliminary, simplified fashion. The objective of the testing was to establish a basis for comparison with previous work by other researchers using the same controller. Therefore, the tests performed were similar to those done previously. Test results were analyzed with the control systems CAD package, MATRIXX [23].

1.5 Contribution and Summary.

The main contributions of this research are summarized below.

- A three axis force and moment sensing system has been integrated with the existing AFIT Robotics Laboratory equipment.
- A compliant motion control environment has been established.
    - The environment incorporates three different digital processors, the force sensor, and a PUMA 560 robot arm into a hierarchial control system.
    - Original techniques have been developed for transforming force sensor data to interface force, and also for eliminating the tool weight from the force sensor measurements.

  - The environment uses modular software and has a high degree of commonality with the existing ARCADE control environment.

- A physical environment for testing compliant motion control algorithms has been constructed. The environment supports future demonstrations of robotic aircraft refueling.

- A preliminary version of an impedance control law has been implemented in the compliant motion environment using links 2 and 3 of the PUMA arm.

  - The implementation on the PUMA is the first known use of this compliant motion control law on a vertically articulated, gear driven manipulator.

  - The implementation is structured to facilitate expansion to a more general impedance control law using all six PUMA links.

  - The implementation uses parallel processing to allow high speed calculation of control torque commands.

- The ability to accurately determine interface force from the force sensor data has been quantified.

- The execution times of the impedance control algorithm's major routines have been measured.

- Tests of the compliant motion controller, emulating those performed by Hogan [20], have been completed.

  - The impedance controller successfully used active compliance to reduce the interface force between the manipulator and the environment.

  - The impedance controller's trajectory tracking performance requires improvement. Performance can be improved by:

    * Including a commanded end-effector velocity term in the control algorithm.

        - Using smaller sample times.
        - Improving the friction compensation model used for the PUMA arm.
    - The controller's inability to emulate Hogan's experimental results is attributed to the complexity of the PUMA arm's dynamics versus those of Hogan's direct-drive manipulator [20].

These contributions have laid a foundation for further broad-based compliant motion research in the AFIT Robotics Laboratory. The compliant motion environment can be augmented by machine vision systems to create an intelligent robotic system capable of realistically demonstrating robotic aircraft refueling.

1.6 Organization.

The remainder of this thesis is organized into five chapters. Chapter Two reviews pertinant past research in force sensing and compliant motion. Chapter Three discusses the impedance control law theory, the assumptions made in simplifying the impedance control law, and the methodology for scaling, calibrating and transforming the force sensor data to interface force. Chapter Four describes the hardware and hierarchial computer control systems used to establish the compliant motion environment. Chapter Five presents results from testing the preliminary version of the control law, while Chapter Six presents conclusions and suggested areas of further research. Three appendices supplement the main body of the report. Appendix A presents the detailed mathematical development of the full impedance control law. Appendix B provides background information on aircraft refueling and a proposed robotic refueler concept. Appendix C contains brief descriptions of the new or modified software used in the compliant motion controller. Complete listings of software developed or modified for this thesis are contained in AFIT Robotic Systems Laboratory Report No. 3 [14].

II. Literature Review and Control Law Selection

2.1 Force Sensor Design and Calibration.

Shimano and Roth present an excellent summary of the design and calibration of force sensors in [50]. While this thesis effort did not require design of a force sensor, the information presented below provides an understanding of how the sensor functions and the mathematics involved.

The JR3 force sensor used for the robotic refueling research is a wrist mounted, strain gage force sensor. Shimano and Roth provide a detailed discussion of the Scheinman force sensor (see Figure 2.1), which is similar to the JR3 sensor. The Scheinman sensor uses a solid aluminum cross as the load carrying structure. The cross is instrumented with eight pairs of strain gages. Shimano and Roth indicate the high stiffness of the cross insures the sensor has little effect on the overall manipulator natural frequency, and also allows accurate end-effector positioning. Also, because the sensor is stiff it does not add any passive compliance to the manipulator. Use of strain gages in pairs prevents changes in calibration due to temperature. The low hysteresis design of the sensor minimizes non-linearities in the force-strain relationship [50, p. 341].

A very straightforward method is used for converting strain gage readings into the three forces and three moments about some user defined coordinate frame. The essential equation is:

$$\vec{F} = RDC\vec{\epsilon} \tag{2.1}$$

where $\vec{\epsilon}$ is the vector of strain gage measurements, $\vec{F}$ is the vector of forces and moments, and C is a calibration matrix decoupling the strain gage measurements into forces and moments about three orthogonal coordinate axes. The matrices R and D are coordinate transformations allowing the user to define a coordinate

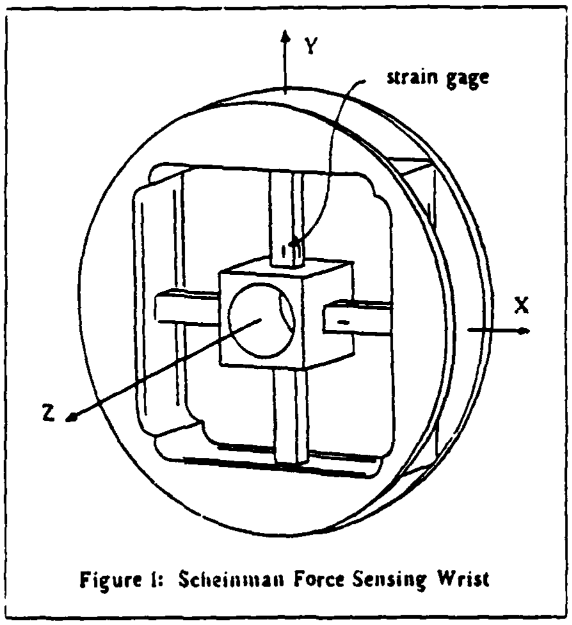


Figure 2.1. Scheinman Wrist Force Sensor. [50, p. 341]

frame which is rotated (using R) and displaced (using D) from the coordinate frame of the strain gage measurements [50, p. 342][1].

The ability to measure forces in a user defined frame is advantageous. Typically, the forces of interest are those at the contact point of the tool and the environment, and not those at the juncture of the tool and the force sensor. Also, force and position information are frequently used together in the compliant motion control law. To do this it may be necessary to have force and position information in the same coordinate frame.

Some researchers have noted some problems with wrist mounted force sensors. All those problems are a result of the dynamics of whatever tool or gripper is mounted between the force sensor and the environment. Salisbury identified a need to eliminate the gravitational force exerted by the tool from the sensor measurements of the contact force [49, p. 384]. Salisbury, as well as Raibert and Craig noted the inertial forces resulting from the linear and angular accelerations of the hand or tool will also appear in the sensor force measurements; although these will be small at low speed, they could be problems at higher speeds and may require additional dynamics calculations to eliminate them from the sensor readings [49, p. 384],[47, p. 378]. Raibert and Craig also noted high frequency oscillation in their force sensor readings from the spring-mass behavior of the tool interacting with the environment, but they were able to suppress this noise using an analog filter [47, p. 381].

---

[1] The D matrix used in Eqn. 4 of [50] is incorrect. The signs of the d_i terms should be reversed.

2.2 Compliant Motion Control Laws.

As mentioned in Section 1.3.1.3, a wide variety of compliant motion techniques have been proposed. Three promising techniques were chosen as possible candidates for the compliant motion control law implemented in this study. They are:

- Stiffness Control
- Hybrid Control
- Impedance Control

Stiffness control introduces the fundamental concept of stiffness as a parameter in compliant motion control. Hybrid control is a conceptually straight forward technique, where task oriented coordinates are identified and divided into force or position control directions. Impedance control is a very promising technique since it allows dynamics based control of the robot during both contact and non-contact portions of the task. These three control techniques are discussed further in Sections 2.2.1, 2.2.2 and 2.2.3, as well as in Chapter Three.

2.2.1 Review of Stiffness Control. Salisbury was the principal developer of the stiffness control technique. He presented the fundamental concept of stiffness, his stiffness control law, and the results he achieved in performing a typical assembly task in [49]. Stiffness control is a simple, but fundamental technique, and the concept of stiffness is fundamental to the more involved techniques of hybrid and impedance control.

Salisbury modeled the interaction of the end-effector with the environment as a six degree of freedom spring (three degrees in position and three degrees in orientation). The spring constants, or stiffness, are the rate constraint forces change with respect to position and orientation errors in the end-effector [49, pp. 383–384]. For cartesian coordinates the relationship is

$$\vec{F} = K\delta\vec{x} \tag{2.2}$$

where $\vec{F}$ is the 6×1 constraint force vector, K is the 6×6 stiffness matrix, and $\delta\vec{x}$ is the error in position. The error is the difference between the actual, constrained position, and the nominal, unconstrained position

$$\delta\vec{x} = \vec{x} - \vec{x}_o \tag{2.3}$$

where $\vec{x}$ and $\vec{x}_o$ are the constrained and unconstrained positions, respectively [49, p. 384].

Understanding the significance of the stiffness matrix is key to an appreciation of Salibury's concept, as well as many other forms of compliant motion control. Stiffness is the inverse of compliance. The elements of the stiffness matrix, K, represent the position accuracy the user desires to achieve along each of the six degrees of freedom. A high stiffness value indicates a strong desire to follow the nominal trajectory in a particular direction (low compliance), and if a constraint is encountered a large interaction force will develop. A low stiffness value indicates the end-effector does not have to follow the trajectory too closely in the given direction and the end-effector will move to accommodate any constraints encountered (high compliance), thereby minimizing the interaction forces [49, p. 383]. Diagonal K

matrices decouple the manipulator motion by causing errors in position along a particular axis to only effect the subsequent motion or force along the same axis. Non-diagonal K matrices can be used to couple motion and forces between different coordinate axes. For example, with a non-diagonal K matrix, position error in the y direction can be used to generate a force in the x direction.

In modeling the interaction as a six dimensional linear spring, Salisbury assumed there were no higher order dynamics terms due to non-linear elastic behavior in the environment or the manipulator. He also did not allow for any viscous damping or stiction at the interface between the end-effector and the environment. These assumptions are similar to those made by Hogan and Kazerooni in their development of impedance control (see Section 2.2.3 and Chapter Three).

Salisbury presented his stiffness control law as [49, pp. 384–385]

$$\vec{T} = \vec{T}_c + G\delta\vec{T} + K_v C_{II}\delta\dot{\vec{\theta}} + V_o \mathrm{sgn}\left(\dot{\vec{\theta}}\right) + \vec{C}_I \tag{2.4}$$

where

$\vec{T}$ is the vector of torques applied to each joint

$\vec{T}_c$ is the vector of commanded torques for each joint

G is the diagonal matrix torque compensation function (e.g. lead-lag filter, with gain K_f, zero at $s = -a$, and pole at $s = -b$.)

$\delta\vec{T} = \vec{T}_c - \vec{T}_s$ i.e. the error between the commanded torque and the sensed torque

K_v is a derivative gain diagonal matrix

C_{II} is the instantaneous inertia of the manipulator

$\delta\dot{\vec{\theta}} = \dot{\vec{\theta}}_o - \dot{\vec{\theta}}$ i.e. the error between the commanded and actual joint velocities

V_o is a vector of Coulomb friction coefficients for each joint

$\vec{C}_I$ is a vector of gravitational torques on each joint

The manipulator's Jacobian is used to convert cartesian forces at the end-effector to torques at the joints, for example

$$\vec{T}_s = J^T \vec{F}_s \tag{2.5}$$

where

$\vec{F}_s$ is the sensed force on the end-effector

$\vec{T}_s$ is the vector of joint torques due to $\vec{F}_s$

J is the manipulator Jacobian

The form of the manipulator Jacobian varies depending on the choice of orientation angles for the position vector [43, pp. 10–11]. However, the Jacobian is commonly found as the vector derivative of the forward kinematics function, $\vec{L}(\vec{q})$, with respect to the joint position vector, $\vec{q}$,

$$J = \frac{d\vec{L}(\vec{q})}{d\vec{q}} \tag{2.6}$$

(The Jacobian is discussed in greater detail in Section 3.1.6.2.)

Using the relationship of Eqn. 2.2 and another Jacobian relationship

$$\delta\vec{x} = J\delta\vec{\theta} \tag{2.7}$$

Salisbury wrote the commanded torque (T_c) as a function of the joint position error ($\delta\vec{\theta}$)

$$T_c = J^T K J \delta\vec{\theta} + T_B \tag{2.8}$$

where he added a bias torque, T_B, to allow a minimum, or bias, force (F_B) to be applied by the end-effector

$$\vec{T}_B = J^T \vec{F}_B \tag{2.9}$$

Algebraic substitution into Eqn. 2.4 produced the complete control law

$$\begin{aligned} T = {} & (1+G)\left[J^T K J(\vec{\theta}_o - \vec{\theta}) + G J^T F_B\right] - J^T F_s \\ & + K_v C_{II}(\dot{\vec{\theta}}_o - \dot{\vec{\theta}}) + V_o \mathrm{sgn}(\dot{\vec{\theta}}) + C_I \end{aligned} \tag{2.10}$$

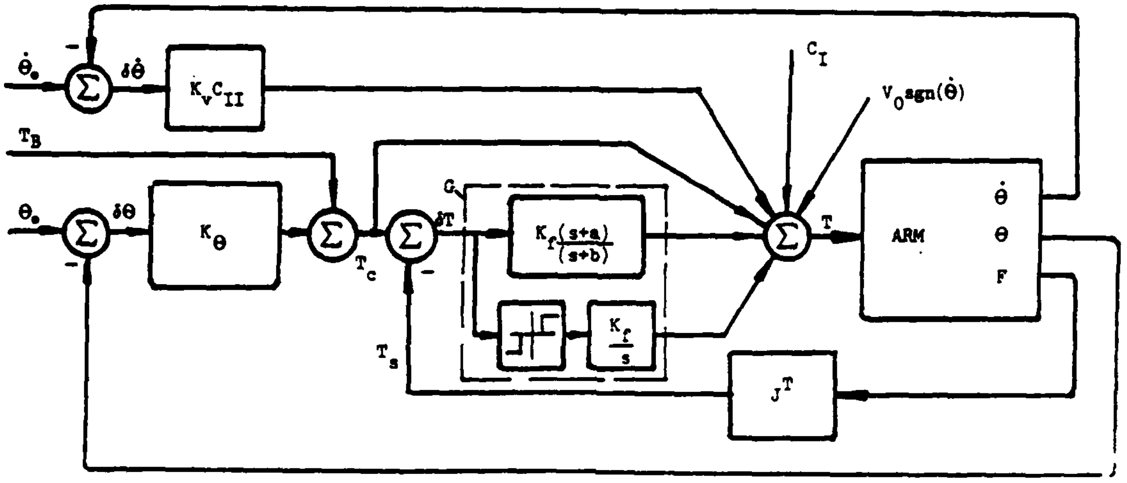

$\dot{\theta}_e$
$\delta\dot{\theta}$
$K_v C_{II}$
T_B
θ_e
$\delta\theta$
K_θ
T_c
G
δT
$K_f \frac{(s+a)}{(s+b)}$
$\frac{K_f}{s}$
T_s
C_I
$V_0 \mathrm{sgn}(\dot{\theta})$
T
ARM
$\dot{\theta}$
θ
F
J^T

Figure 2.2. Stiffness Control System. [49, p. 385]. Note: $K_\theta = J^T K J$.

The control law is summarized in block diagram form in Figure 2.2. The control law is fundamentally a proportional-derivative feedback loop with the addition of friction and gravity compensation, as well as the inclusion of force feedback. Gravity and friction are included because they are significant dynamic forces, even when the manipulator is moving at low speeds. The instantaneous inertia, C_{II}, is included with the derivative control term (K_v term) to adjust the damping for the varying manipulator inertia, and possibly varying payload mass. The compensator function, G, includes lead-lag and integral compensators to improve the transient force response and drive force errors to zero. Salisbury included a deadband-limit function with the integral compensator to prevent limit cycling and restrict the integrator's response to impact forces [49, pp. 384–385].

Salisbury implemented his stiffness control law on a Stanford-Scheinman arm using a PDP 11/45 with cache memory as the control computer. To ease the computation burden he updated the Jacobian at 12 Hz while running the rest of the controller at 60 Hz. Computation of the control torque values for the arm required 5.3 milliseconds [49, p. 385]. Using this manipulator he was able to successfully complete peg-in-the-hole type insertion tasks.

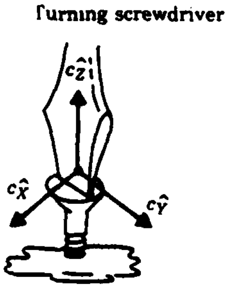
Salisbury suggested zeroing out the weight of the tool on the wrist force sensor just before making contact with the environment [49, p. 384]. Resetting the force sensor in this manner "eliminated" the weight of the tool from the force sensor readings. It is an effective technique for tasks where the tool maintains approximately the same orientation throughout the job. However, for tasks where the tool makes large changes in orientation this technique will not be accurate. A more versatile technique would be to calculate and subtract the gravitational force exerted by the tool based on its measured orientation.

2.2.2 *Review of Hybrid Control.* Hybrid control is a simple, straightforward approach to compliant motion control. Partly as a result of its intuitiveness, hybrid control has become a very popular method for implementing a compliant motion controller. With hybrid control some coordinate axes are controlled based on position while others are controlled based on force. Hybrid control is a top-level, architectural approach, and does not require use of a particular control law, although typically a stiffness-based controller is implemented.

A number of researchers have written extensively about hybrid control. The fundamental research on this subject was done by Mason, and Raibert and Craig. Mason provided the formalized theory for establishing task specific constraint coordinate frames [42]. Raibert and Craig expanded on this theory to develop the hybrid control architecture, and have experimentally implemented a stiffness-based control law within the architecture. Raibert's and Craig's research is documented in [47], and also in [6].

The unique feature, fundamentally establishing the concept of hybrid control, is the use of *constraint surfaces* and *constraint frames* to define the compliant motion task. Mason defined a constraint surface (also called a C-surface) as a

> ...surface in configuration space...which allows only partial positional



Natural constraints

$v_x = 0$ $f_y = 0$
$\omega_x = 0$ $n_z = 0$
$\omega_y = 0$
$v_Z = 0$

Artificial constraints

$v_y = 0$ $f_x = 0$
$\omega_z = \alpha_2$ $n_x = 0$
$n_y = 0$
$f_z = \alpha_3$

Figure 2.3. Natural and Artificial Constraints for Turning a Screwdriver. [6, p. 261]

> freedom (*author's note:* and only partial force freedom). Freedom of motion occurs along C-surface tangents, while freedom of force occurs along C-surface normals [42, p. 361].

A set of *natural constraints* arise from the partial freedoms of motion and force existing on the constraint surface. The natural constraints on motion (velocity) exist in directions normal to the constraint surface, while the natural constraints on force exist in directions tangent to the constraint surface [42, p. 364],[6, p. 260]. For example, in Figure 2.3 a screwdriver rotates a screw. If the screw slot is frictionless then naturally no force can be exerted in the $c_{\hat{Y}}$ direction. Similarly, motion is constrained in the $c_{\hat{X}}$ direction by the side of the screw slot, so velocity in the $c_{\hat{X}}$ direction is naturally zero. However, in general the natural velocity constraints need not be zero [42, p. 366]; for example, if the screw were mounted on a moving conveyer belt, then the velocity in the $c_{\hat{X}}$ direction would be the velocity of the conveyer belt.

The conjugate to natural constraints are *artificial constraints*. They also exist as constraints on motion and force along the constraint surface, but they are orthogonal to the natural constraints. Therefore, artificial motion constraints lie along constraint surface tangents, and artificial force constraints lie along constraint surface normals [42, p. 364],[6, p. 260]. For the screwdriver example, force in the $c_{\hat{z}}$ direction is artificially set at some value, α_3, sufficient to insure constant contact between the screw and screwdriver. The velocity in the $c_{\hat{y}}$ direction is artificially set to zero to prevent the screwdriver from moving out of the slot. When implementing the hybrid controller, the artificial constraints are the controlled variables in the feedback control loops [6, p. 260].

For a complex task, the constraint frames and constraint sets must be time varying [42, pp. 371–372]. The naturally constrained variables are monitored, and changes are used to indicate the task status, as well as to initiate changes in constraint frames and constraint sets [6, pp. 262–263]. In the screwdriver example, a sudden increase in the moment about the $c_{\hat{z}}$ axis, n_z, indicates the screw has been driven all the way in.

The process of synthesizing a hybrid control task strategy begins with making assumptions about the task in order to establish the constraint surface and coordinate frame. The constraint surface and task assumptions generate a set of natural constraints. Next, artificial constraints are established orthogonal to the natural constraints, so as to cause the manipulator to move along the desired trajectory. A compliant motion control law is required to transform these artificial constraints into the desired trajectory [42, p. 362,372].

The above discussion of constraint surfaces and constraints is a simplified summary of Mason's research [42]. Mason presented his theory in a very generalized and rigorous manner. In addition to establishing the fundamental theory, he discussed techniques for coordinated compliant motion control of multiple manipulators. He also described techniques for handling coupling between natural and artificial constraints, as occurs when friction is present (because friction is a tangential force dependant on a normal force) [42, pp. 367–370].

Figure 2.4 shows the general architecture of a hybrid controller. Since the desired trajectory is described in the constraint coordinate frame (where it is more

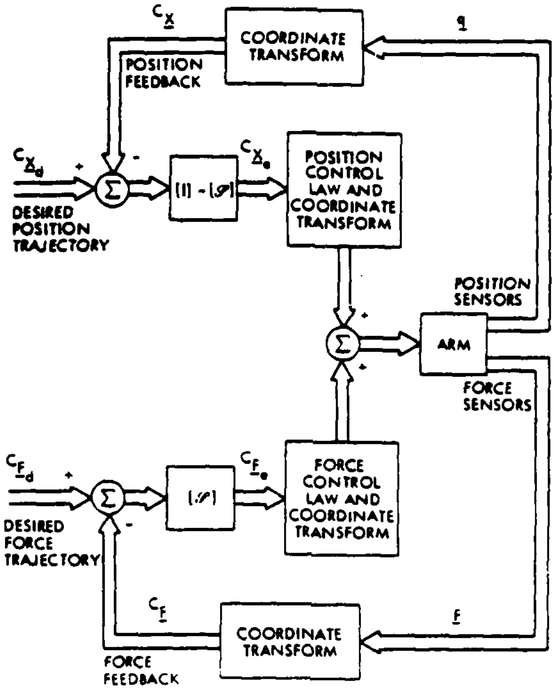

$^{C}\underline{x}$
COORDINATE TRANSFORM
$\underline{q}$
POSITION FEEDBACK
$^{C}\underline{x}_d$
+
-
DESIRED POSITION TRAJECTORY
Σ
$[I]-[S]$
$^{C}\underline{x}_e$
POSITION CONTROL LAW AND COORDINATE TRANSFORM
POSITION SENSORS
+
Σ
+
ARM
FORCE SENSORS
$^{C}\underline{F}_d$
+
DESIRED FORCE TRAJECTORY
Σ
-
$[S]$
$^{C}\underline{F}_e$
FORCE CONTROL LAW AND COORDINATE TRANSFORM
$^{C}\underline{F}$
COORDINATE TRANSFORM
$\underline{F}$
FORCE FEEDBACK

Figure 2.4. Hybrid Control Architecture. [47, p. 377]

readily conceived), the joint position and force sensor measurements must be transformed into the constraint coordinate frame before an error signal can be generated [47, p. 377].

The choice of control laws for the position and force control loops is left to the user. The user must consider his need for performance versus his computer's ability to quickly complete complex calculations. The position control loop can be as simple as a proportional-derivative (PD) or a proportional-integral-derivative (PID) controller, or it could be as complex as a full feedforward dynamics control law [47, pp. 378–379]. Likewise, the force control loop can involve only a proportional or proportional-integral controller, or it could involve a more complex form of compliant motion control, such as the stiffness based approach of Eqn. 2.2 [47, p. 379], [6, p. 273–277].

The output commands from the position and force control laws will be in terms of constraint frame coordinates. These commands must be transformed into joint space before they can be passed to the arm's actuators. The type of transformation will depend on the form of the constraint frame, but may involve use of the manipulator Jacobian inverse [47, p. 377, 380].

The controller is a hybrid since it combines position and force control loops into a single compliant motion controller. "While each degree of freedom in C (the constraint frame) is controlled by only one loop, both sets of loops act cooperatively to control each manipulator joint [47, p. 378]." The choice of which degrees of freedom in the constraint frame are force controlled, and which are position controlled, is made from the artificial constraints established in the task strategy. The decision is implemented through the selection matrices, S, as shown in Figure 2.4 [47, p. 377]. For the screwdriver example,

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.11}$$

since artificial force constraints exist on f_x, f_z, n_x, and n_y.

Raibert and Craig implemented their hybrid controller using a Scheinman wrist force sensor on a Stanford-Scheinman arm controlled by a General Automation SPC-16/85 minicomputer. They used a PI force control law and PID position control law [47, pp. 377–378]. Experiments were conducted testing the manipulator's ability to: maintain a constant force on a table moving in the xy-plane; respond to step, ramp and sinusoidal position and force commands; and insert a peg in a hole with only .001 inch clearance. The manipulator successfully completed all the tests. Limit cycling of the constraint force was observed and attributed to Coulomb friction, a result subsequently analytically confirmed by Townsend and

Salisbury [54]. An oscillatory response in the constraint force occurred when the manipulator was commanded to follow a table moving at a slow, constant speed. Force oscillation amplitude was greatest at the beginning and end of the table motion, and while the oscillation was significant, contact with the table was not lost. When following an applied force ramp input, the manipulator's actual force response again experienced small amplitude oscillation, apparently random in nature. Throughout the course of their experiments Raibert and Craig did not see any evidence of coupling between errors in the position and force control loops [47, pp. 380-381].

Raibert and Craig noted hybrid control is similar to the compliant motion control technique developed by Paul and Shimano. According to Raibert and Craig, Paul and Shimano's technique assigned control of a force constraint axis to a specific manipulator joint for each time increment of the trajectory. The technique is non-optimal since it results in position and force errors after every increment and correcting these errors requires the trajectory to be adjusted. Raibert and Craig implied their hybrid approach is more efficient since adjustment of the trajectory after each increment is not necessary [47, p. 378].

Khatib has also developed a compliant motion control technique similar to hybrid control. The principal difference is Khatib's use of the *operational space* in calculating the dynamics of the end-effector [30, p. 44]. The operational space coordinates are essentially the cartesian coordinates of the end-effector frame rotated to align the end-effector coordinate frame with the constraint frame [30, p. 44]. Use of the operational space allows a unified approach, because the problem is both defined and controlled in the constraint space. However, the approach appears to be about as computationally complex as hybrid control. A more significant contribution is Khatib's extension of operational space concepts to compliant motion control of redundant manipulators [30, pp. 49–51].

Khatib implemented his operational space force controller on a PUMA 600 controlled by four National Semiconductor 32016 microprocessors in the COSMOS control system. The position and force control loops both employed PD feedback control laws. He was able to demonstrate the compliant assembly tasks of surface contact and sliding along a surface, as well as insertion [30, pp. 49, 52]. These results are documented in detail in [4].

Hybrid control offers a number of advantages for compliant motion control. The technique is conceptually easy to understand, does not restrict the user to a particular control law, and allows independent control of force and position. Unfortunately, for the robotic refueling application hybrid control has some drawbacks. It can be difficult to establish constraint surfaces and the corresponding natural and artificial constraints. This is particularly true for the robot refueler because the problem is not highly structured, i.e. the aircraft and the robot are not always in the same relative position and orientation for each individual refueling. As a result, the constraint frame must be redefined, or the position and orientation of the robot relative to a known constraint frame on the aircraft must be determined. Additionally, hybrid control requires switching of the selection matrices, S, as different parts of the task are performed. Using different S matrices might cause stability problems because of the possibility of dramatic changes in torque commands as the constraints are changed. The need to switch constraints as the end-effector impacts a surface may also lead to stability problems if the end-effector bounces and loses contact with the surface. In this case the constraints may require the manipulator to exert a certain normal force on the surface, but if the end-effector bounces off the surface it will not exert any force. The resulting large force error and corrective command could lead to an even larger bounce, with the manipulator eventually becoming unstable.

2.2.3 Review of Impedance Control[2]. Hogan and Kazerooni have been advocates of a compliant motion control technique called *impedance control.* Hogan established the groundwork for considering a manipulator as a mechanical impedance, and also developed a control law for an impedance controller [17] [18] [19]. Kazerooni similarly justified the use of impedance control and developed a model for the desired system impedance in the frequency domain [26], [27].

With impedance control, the manipulator's non-linear dynamics are changed by the action of the controller to be the linear dynamics of a simple, second order, mass-spring-damper system. The technique for developing the control law is similar to pole placement design methods where the desired dynamics and the actual dynamics are compared algebraically to determine the form of the controller [18, p. 11]. Hogan presented the control law as

$$\begin{aligned}\tau_{act} = {} & I(\theta)J(\theta)^{-1}M^{-1}K\left[x_o - L(\theta)\right] + S(\theta) + I(\theta)J(\theta)^{-1}M^{-1}B\left[v_o - J(\theta)\omega\right] \\ & + V(\omega) + I(\theta)J(\theta)^{-1}M^{-1}F_{Int} - J(\theta)^T F_{Int} \\ & - I(\theta)J(\theta)^{-1}G(\theta,\omega) + C(\theta,\omega) \end{aligned} \tag{2.12}$$

where

θ is the measured joint position vector

ω is the measured joint velocity vector

F_{Int} is the interface, or constraint, force vector measured by a force sensor at the end-effector

τ_{act} is the commanded actuator torque vector

$I(\theta)$ is the manipulator inertia tensor

$J(\theta)$ is the manipulator Jacobian matrix

---

[3] Because impedance control is explained in detail in Chapter Three, this review will focus on a primarily qualitative description of other researchers' work in the field of impedance control.

$M(\theta)$ is the target dynamics inertia tensor for modelling the system as a simple rigid body

K is the target dynamics stiffness matrix, designed for accomplishing the task

x_o is the unconstrained position vector of the end-effector in cartesian coordinates (also known as the virtual position or displacement)

$L(\theta)$ is the forward kinematics function transforming the joint position vector into the cartesian position vector; the term represents the end-effector's constrained position

$S(\theta)$ is the vector of gravitational torques acting on each joint

B is the target dynamics damping matrix, chosen to stabilize the target dynamics

v_o is the unconstrained velocity vector of the end-effector in cartesian coordinates (also known as the virtual velocity)

$V(\omega)$ is the velocity dependant friction torque vector

$G(\theta,\omega) = \left[\frac{d}{d\theta}(J(\theta)\omega)\right]\omega$ a vector term resulting from use of the product rule in the differentiation of the kinematics

$C(\theta,\omega)$ is the vector of Coriolis and centrifugal torques

The values for x_o and v_o represent the virtual trajectory for the end-effector. The virtual trajectory is the path the manipulator would take if no environmental motion constraints existed. When the end-effector meets a constraint, and can no longer follow the virtual trajectory, the difference between the virtual and the actual trajectory will determine the constraint forces on the manipulator. Thus, the constraint forces can be tailored through proper design of the virtual trajectory and selection of the stiffness matrix (see Section 2.2.1) [18, p. 9].

Hogan's impedance control [illegible] included a fairly complete dynamic model of the manipulator. Coriolis, centrifugal, inertial, friction, and gravity effects are all incorporated. It required the position, velocity, and force information in a cartesian coordinate frame. The control law can be thought of as a PD controller augmented by feedforward dynamic compensation and force feedback. The PD and force feedback gains are modulated by the manipulator inertia.

The development of the control law assumed higher order vibrational dynamics did not exist in the environment or the manipulator. In order for the kinematics function, $L(\theta)$, to be accurate there must be no passive compliance in the manipulator. All the constraint forces are assumed to be linear functions of the difference between the virtual trajectory and the desired trajectory [18, p. 9–10]. This last assumption, while accurate for linear forces (such as simple springs and dampers), does not include non-linear forces such as stiction or certain non-linear elastic forces resulting from deformation of the robot or environment.

For simplicity, both Hogan and Kazerooni desired the manipulator to behave as a second order linear system.

$$\frac{X(s)}{F_{Int}(s)} = \frac{1}{Ms^2 + Bs + K} \tag{2.13}$$

where M, B, and K are as defined above, s is the Laplace variable, and

$F_{int}(s)$ is the interface or "disturbance" force due to interaction with the environment in the Laplace domain

$X(s)$ is the response of the target dynamics in the Laplace domain

Using the model of the desired dynamics, Kazerooni developed two theorems regarding the stability of the manipulator. Just like a scalar second order system, the eigenvalues of the denominator of Eqn. 2.13 will determine the overall stability of the manipulator [26, p. 88]. Kazerooni's first theorem defines *sufficient but not necessary* conditions for stability

> If M,B, and K are real and symmetric, positive definite matrices, then the target dynamics are stable, and if B and K are symmetric, non-negative definite matrices, then the target dynamics will be

> marginally stable. If K and/or B are symmetric, positive, semi-definite matrices, then some or all eigenvalues will be on the imaginary axis. (These cases are considered unstable.) [26, p. 88].

Kazerooni indicated just because the target dynamics are stable does not mean the actual combined manipulator-environment dynamics will be stable. The overall stability of the manipulator and environment are governed by *sufficient* conditions expressed in a second, more complex theorem. Essentially, the actual combined system dynamics are required to behave exactly like the target dynamics, and M, B, and K must be symmetric, positive definite matrices [26, pp. 88–89]. Assuming well known dynamics, it is possible to design a controller that will modify the system behavior to be very close to the desired dynamics. Kazerooni presented a complex eigenstructure assignment technique for developing such a controller in [27].

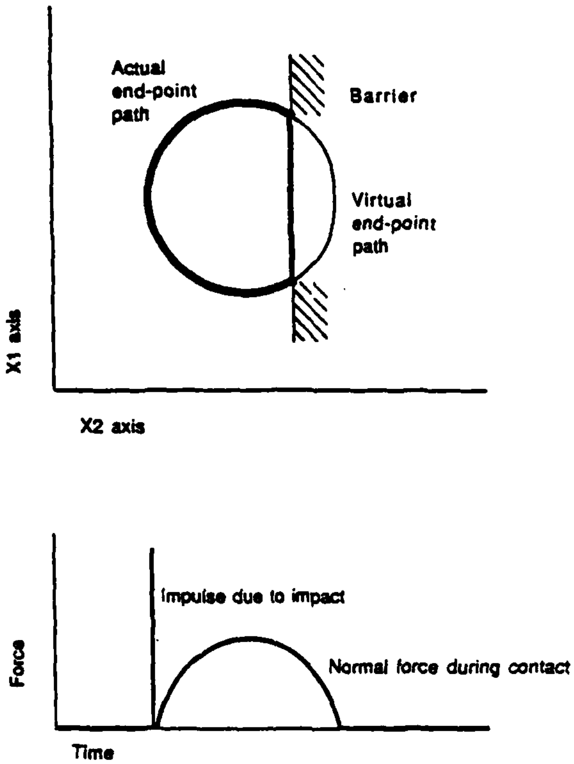
The results of Kazerooni's first theorem have been experimentally confirmed by Lawrence [31, p. 1188]. In addition, Lawrence considered the effect on stability of a computation time delay factor (e^{-st_d}) in the feedback loop of the controller. He determined the time delay placed lower bounds on the stable range of values for B and K.

> Larger time delays require larger amounts of damping for a given cartesian stiffness. Large damping in the cartesian behavior can significantly increase interaction forces unless the motion is extremely slow, and (large damping therefore) limits the 'softness' or 'dexterity' of the manipulator [31, p. 1188].

Kazerooni and Tsay have noted stable motion is not possible without some amount of compliance between the manipulator and the environment [29, p. 1170],[28, p. 330]. In the presence of a stiff environment, and when using a robot with little inherent passive compliance, it is the job of the controller to actively add this necessary compliance.

Hogan demonstrated the use of an impedance control law (Equation 2.12) with a small, revolute, two degree of freedom manipulator using direct drive electric motors. Special low friction bearings were used at both joints, the links had relatively little inertia, and because the manipulator operated in the horizontal plane the motors did not work against gravity. A nine millisecond sample time was used and the arm bandwidth was limited to approximately five Hertz [20, pp. 10500–1051]. The test consisted of moving the end-effector along a circular virtual trajectory. A flat plate was imposed in one portion of the trajectory, causing the actual trajectory to be roughly semi-circular (see Fig. 2.5). The test was run with force-feedback (active compliance), and without force-feedback (only the system's inherent passive compliance). In both cases the system was stable, even though the environment and force sensor were very stiff. The use of the impedance controller created a much smoother (less bouncing) transition from unconstrained to constrained motion (see Fig. 2.6). The controller tracked the virtual trajectory when it was not constrained by the plate, and moved along the plate in a stable fashion when it was in contact with the plate [20, pp. 1051, 1053].

Kazerooni also applied impedance control to a specialized manipulator [28]. He built an active remote center of compliance (RCC) device using a five bar closed chain linkage (2 degrees of freedom) actuated by two direct drive motors. The RCC linkage was very small (approximately and two inches by four inches) and lightweight (5.05 lb including motors, force sensor, and payload). As with Hogan's manipulator, the use of direct-drive motors simplified the dynamics by minimizing friction, backlash, and other gear train effects. Kazerooni designed the linkage to uncouple the link dynamics, and he severely restricted the workspace (0.3 inches by 0.3 inches) to force the Coriolis and centrifugal accelerations to be small enough to be ignored. Kazerooni demonstrated a close correlation between the actual dynamic response and the dynamic response predicted by the impedance control model.


Actual
end-point
path
Barrier
Virtual
end-point
path
X1 axis
X2 axis
Impulse due to impact
Normal force during contact
Force
Time

Interface Force vs. Time

Figure 2.5. Hogan's Virtual Trajectory and Ideal Force Response [20, p. 1053].

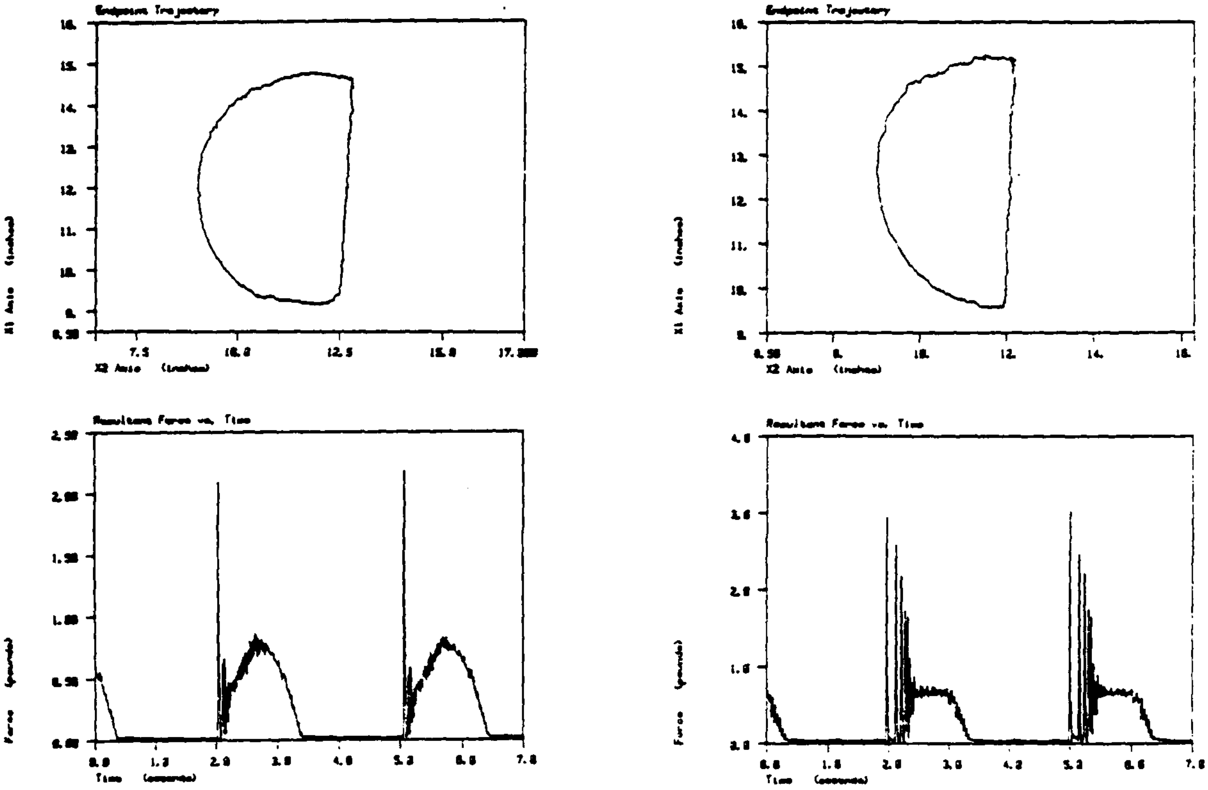


Figure 3. Actual performance of the experimental apparatus on the test task. The path of the end-point is shown on the top. The time history of interface force is shown on the bottom.

Figure 4. Actual performance of the experimental apparatus on the test task when the force feedback was disconnected. The path of the end-point is shown on the top. The time history of interface force is shown on the bottom.

Figure 2.6. Hogan's Impedance Controller Performance [20, pp. 1053–1054].

Impedance control possesses a number of advantageous characteristics motivating its use as the basis for a compliant motion controller. Two of these are

- As a result of the model, both positioning and compliant motion tasks can be controlled with an impedance controller. Therefore impedance control presents a unified approach to robot control [17, p. 6].
- Because a unified approach is used, impedance control allows smooth transitions from unconstrained to constrained motion. For some finite range of velocity it is not necessary to approach contact with environment using a guarded move [20, p. 1052].

Hogan's and Kazerooni's development of impedance control and the associated stability theorems has a number of shortcomings

1. The approach is complex and not intuitively straightforward since it involves modifying the impedance relation to obtain the desired response. Impedance itself is an abstract quantity composed of stiffness, damping, and inertia terms.
2. The requirement to use the inverse Jacobian in the control law presents a minor problem. The computational burden can be overcome through use of high speed or parallel processors in calculation of the control torque. However the possibility of singular points within the arm's workspace remains. Special error routines may be necessary to control the manipulator near these points, or trajectories must be shaped to avoid the singularities.
3. Extensive computations are necessary to determine the Coriolis and centrifugal ($C(\theta,\omega)$) term as well as the $G(\theta,\omega)$ term. Like the Jacobian, the computational burden can be overcome through additional computer power, or the terms could be ignored for low speed motion.

4. The control law also does not include any non-linear dynamic effects inherent in the actuator or drive train. High stiction values and motor cogging have been reported as problems by other researchers doing force control on a PUMA 560 arm [4, p. 70].
5. The control law does not include the effects of arm vibration (e.g. body bending modes) or non-linear elastic forces due to deformation in the robot arm or the environment. For most industrial arms this is not a problem, but for light-weight arms with lower structural stiffness, like the Space Shuttle Remote Manipulator System, this might not be adequate during certain tasks.

Lastly, Hogan's and Kazerooni's demonstrations of impedance control have both occurred on small manipulators specially designed to possess relatively benign dynamic characteristics. A gear driven, vertically articulated arm, such as the PUMA 560, is a much more typical industrial robot. The PUMA is also much more representative of the manipulator configuration best suited for the robotic aircraft refueler. Because the PUMA's dynamics include significant non-linearities (e.g. gear train friction, gravitational torques) it provides a much more challenging implementation of impedance control.

2.3 Additional Compliant Motion Topics.

2.3.1 Effects of Friction. Townsend and Salisbury investigated the effects of friction on force control [54]. They concentrated on *Coulomb friction* and *stiction* (Coulomb friction being the case where static and sliding friction are equal, and stiction being the case where static friction does not equal sliding friction). Viscous friction was not considered, because it was already a well characterized linear effect. Frictional effects on both nearly mass-less and massive friction nodes are discussed, and the results serve to explain limit cycle and some unstable behavior observed on force control systems.

Mass-less friction nodes are those sources of friction resulting from the interaction of relatively light-weight components, such as gears or tendons sliding in a sheath. In contrast, massive friction nodes are characterized by the interaction of heavy components, such as the friction occurring at a joint bearing supporting a heavy link [54, pp. 884–885]. Because mass-less nodes can be analyzed using linear techniques the results obtained are quantitative. Massive nodes exhibit highly non-linear friction effects and must be studied using non-linear techniques. Townsend and Salisbury acknowledged the difficulty and sometimes poor accuracy of these techniques, and the results for massive nodes are principally qualitative.

Townsend and Salisbury used a step input response as the basis for the analysis, since a step input most severely tests a system's stability. Integral and proportional controllers were both examined, although the majority of the analysis concentrates on the integral controller because of its more benign response to a step input. Throughout the analysis they assumed the dynamic response of the friction node would be much faster than the dynamic response of the actuator, a good assumption for most problems. Simulation was used to verify the results of the mathematical analysis [54, pp. 884–888].

Townsend and Salisbury concluded stiction can induce limit cycling in the applied force for both mass-less and massive nodes. The applied force response curve is piece-wise continuous with a "stick" phase (where velocity is zero and stiction is dominant), and a "slip" phase (where velocity is non-zero and sliding, or Coulomb, friction is in effect). For a massive node, stiction also has the effect of reducing the system stability. On the other hand, Coulomb friction has the effect of increasing stability; a result making intuitive sense because Coulomb friction effectively adds a constant damping term to the system dynamics [54, pp. 884–889].

2.3.2 Additional Stability Issues. Stability is an important issue in the design of any control algorithm. An and Hollerbach presented two papers on stability issues for compliant motion controllers. The first deals with dynamic stability in the presence of stiff environments [1], while the second highlights unique kinematic stability problems associated with hybrid force controllers implemented on revolute manipulators [2]. The results of these two papers have important implications for the design of a compliant motion controller.

An and Hollerbach defined two types of instabilities:

Dynamic Instability An instability occurring as the result of interaction between the manipulator and environment dynamics [1, p. 890].

Kinematic Instability An instability arising from the form of the cartesian to joint coordinate transformations required in any compliant motion controller using a cartesian force sensor [2, p. 897].

In their analysis of dynamic instability, An and Hollerbach modeled the interaction of the robot and environment as a mass-spring system controlled by an impedance controller [1, p. 891]. Although they initially began the analysis considering only a single degree of freedom, the results were later shown to be equally applicable to a multiple degree of freedom arm [1, p. 891–892]. The closed loop dynamic equation for a one link robot was

$$m\ddot{x} + \frac{m}{m_d}K_v\dot{x} + \frac{m}{m_d}(K_p + K_e)\,x \;=\; \frac{m}{m_d}K_p x_d + \frac{m}{m_d}K_v\dot{x}_d \qquad (2.14)$$

where

m is the link mass

m_d is the desired link mass, an impedance control parameter specifying the *apparent* mass of the link

x is the cartesian position of the link

x_d is the desired cartesian position of the link

K_v is the velocity gain

K_p is the position gain

K_e is the stiffness of the robot-environment interaction

The left hand side of Equation 2.14 represents the system as a second order harmonic oscillator, while the right hand side is the input forcing function.

A simple root locus analysis of Equation 2.14 demonstrated the system was stable for all positive values of K_v, K_p, and K_e. However, An and Hollerbach indicated an extremely underdamped response will occur if K_v is chosen without considering K_e, since K_e is large for stiff environments or assembly tasks with tight tolerances. In effect, "...force feedback is *very high gain* position feedback [1, p. 891]."

The dynamics of the impedance controlled *mass-spring system* were written as a transfer function $G(s)$. To account for modelling errors and non-linear effects (e.g. friction, gravity) a disturbance term, $E(s)$, was included in the forward path of the control loop. For robust stability, An and Hollerbach required

$$| E(s) | < | 1 + G^{-1}(s) | \tag{2.15}$$

Using frequency domain plots, they were able to show this condition can be met for all frequencies if K_e is small. However, if a stiffer environment was encountered (large K_e), for the same values of K_p and K_v, then the condition of Eqn. 2.15 could be violated, particularly at high frequencies [1, p. 891]. Whitney made a similar conclusion about the difficulty of achieving a stable response in the face of a stiff environment [58, p. 11] and Kazerooni and Tsay have also made the same conclusion (see Section 2.2.3). The important point is: for a robot to be dynamically stable in a stiff environment, the controller must actively change the effective compliance of the manipulator (usually to a lower value).

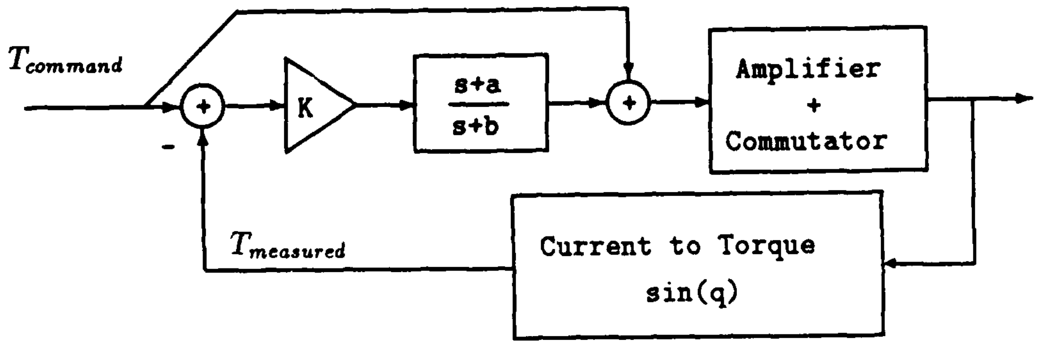
An and Hollerbach indicated kinematic instabilities were not confined to points where the inverse Jacobian was not defined, but that they could occur throughout a wide portion of the workspace. As described below, kinematic instabilities are the result of using certain compliant motion algorithms with certain combinations of the manipulator inertia tensor and kinematic parameters. Because the inertial and kinematic parameters are a function of the arm design (i.e. revolute, polar, cartesian), the possible presence of kinematic instabilities was found to be dependant on the arm type. An and Hollerbach first observed these kinematic instabilities when a hybrid controller was being used to control the MIT Serial Link Direct Drive Arm [2, p. 898].

An and Hollerbach explained kinematic instabilities mathematically by modelling a two link, two degree of freedom, revolute manipulator using linearized dynamics (no gravity, friction, or Coriolis and centrifugal forces). The hybrid force control law developed by Raibert and Craig (see Section 2.2.2) was employed and the system closed loop equations were written. Instability occurred in joint angle regions where the combinations of $M^{-1}(q)$ (the manipulator inertia matrix) and $J^{-1}(q)$ (the inverse manipulator Jacobian) resulted in the closed loop system eigenvalues in the right half s-plane. For the hybrid controller, instability is only possible when force control is commanded along at least one constraint frame axis, however the regions of instability are determined only by $M^{-1}(q)$ and $J^{-1}(q)$ and *contact with the environment is not necessary* [2, pp. 898–899].

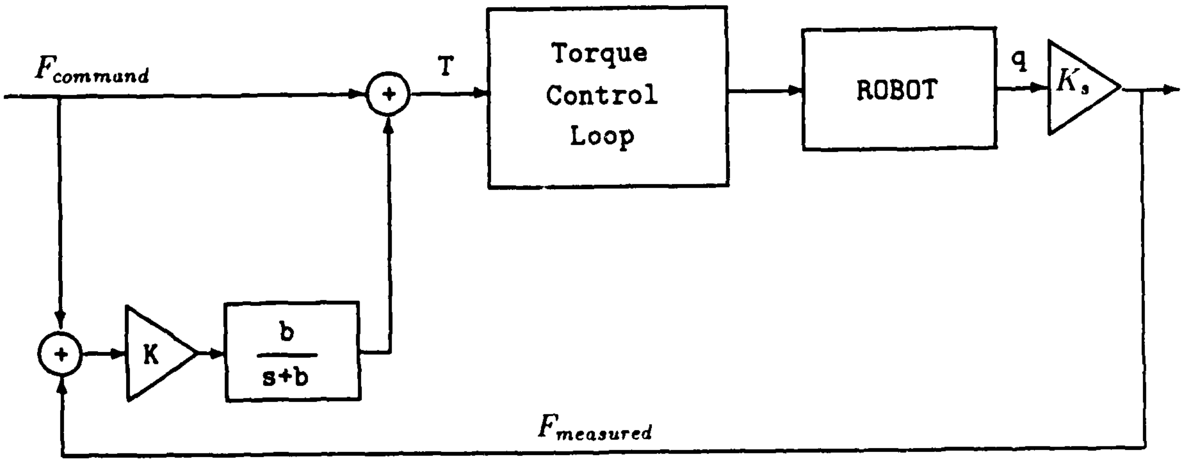
An and Hollerbach applied the same analysis to a polar manipulator using hybrid control, and to revolute manipulators using resolved acceleration force control and stiffness control. Because the polar manipulator has a prismatic link, $M^{-1}(q)$ and $J^{-1}(q)$ have a different format and unstable eigenvalues do not occur [2, p. 901]. The resolved acceleration controllers include an estimate of the actual revolute manipulator inertia matrix, $\hat{M}$, and if the estimate is precise (i.e. $\hat{M} = M$) unstable eigenvalues are prevented from occurring. An and Hollerbach reported good stability robustness for resolved acceleration controllers, although if the error in the estimate is large enough instability can occur [2, pp. 899-900]. The stiffness controller uses the Jacobian transpose, $J^T(q)$, instead of $J^{-1}(q)$, and was proven through simulation to be insusceptible to kinematic instabilities [2, p. 900].

In their experimental research on a revolute arm An and Hollerbach avoided kinematic instabilities by using resolved acceleration control instead of hybrid control [2, p. 901]. Dynamic stability was achieved through the use of an inner and outer feedback loops running at different clock speeds (inner loop = 500 Hz, outer loop = 100 Hz, see Figure 2.7). The inner loop was a relatively high bandwidth servo control loop using joint torque feedback. A compensator was added to this loop to obtain an overdamped first-order type response with limited oscillation amplitude. The outer loop used force feedback from the wrist mounted force sensor, and its bandwidth was limited by a low-pass filter to one Hertz. Force feedback allowed more accurate steady-state control of the constraint forces, and use of a lower bandwidth insured system stability was not affected [1, pp.893–894].

An and Hollerbach tested their compliant motion controller using both one and two links of the MIT Serial Link Direct Drive Arm. In both cases, the link(s) operated in a plane normal to the gravitational force. Response to step and sine wave force inputs was measured, as well as the manipulator's ability to follow an oscillating surface with constant force. An aluminum surface provided a stiff physical environment. The inner joint torque control loop was tested separately, and although a good transient response was observed, a steady-state force error of approximately 6% existed. When the outer force control loop was added, no steady-state error was observed. They concluded the use of an outer force feedback loop with the inner joint torque feedback loop improved steady-state force error without degrading the transient response. The robot's ability to accurately follow a sine wave input or oscillating surface degraded as the input frequency increasingly


$T_{command}$
-
K
$\frac{s+a}{s+b}$
Amplifier + Commutator
$T_{measured}$
Current to Torque
sin(q)

INNER TORQUE CONTROL LOOP


$F_{command}$
T
Torque Control Loop
ROBOT
q
K_s
K
$\frac{b}{s+b}$
$F_{measured}$

OUTER FORCE CONTROL LOOP

Figure 2.7. Inner and Outer Control Loops. [1, pp. 893–894]

exceeded the bandwidth of the outer loop [1, pp. 893–894], [2, pp. 901–902].

An and Hollerbach's chief contributions were the discovery and explanation of kinematic instabilities associated with hybrid control on revolute arms, and the presentation of a simple technique for creating an accurate, stable force controller. They justified the use of step and sine wave inputs as a realistic measure of compliant motion controller performance. Unfortunately, their experimental results only considered at most a two link planar manipulator (no gravitational dynamics). By using a direct drive arm they also avoided many of the complex non-linear effects (e.g. backlash, cogging, friction) associated with current, gear driven, industrial manipulators.

2.4 *Control Law Selection.*

The stiffness, hybrid, and impedance control compliant motion techniques were considered for implementation in the compliant motion environment developed for this thesis. Information gathered in the literature review and discussed in Section 2.2 and 2.3.2 were used to make a choice among the three concepts.

Hybrid control initially appeared attractive because of its simplicity. However, hybrid control divides a task along orthogonal artificial and natural constraint frames, and both force and position cannot be simultaneously controlled along the same axis. This poses a problem for compliant motion tasks involving a moving constraint surface. The robot refueling problem needs a moving constraint surface because the refueling port will move as the aircraft settles on its landing gear during the loading of several thousand pounds of fuel.

More importantly, the research of An and Hollerbach indicated kinematic instabilities could occur throughout the workspace of a revolute manipulator using hybrid control. The AFIT Robotics Laboratory has only revolute manipulators available for the testing of robotic control algorithms. As a result, hybrid control was not considered a viable alternative for implementation in this thesis.

Salisbury's stiffness control law and Hogan's impedance control law are similar. Both contain damping terms dependant on velocity errors and manipulator inertia. Neither require the inverse kinematics in the control loop (a computational difficulty). They also include two of the dominant dynamic effects on a heavy, gear driven manipulator: friction and gravity. Each require the Jacobian and transpose Jacobian to transform cartesian quantities into joint space quantities. Both have been demonstrated in at least a preliminary fashion.

Stiffness control allows a minimum contact force (F_B) to be specified by the user explicitly. In impedance control contact forces can only be specified implicitly through clever design of the virtual trajectory and stiffness matrix. Also, stiffness control does not require the inverse Jacobian, while impedance control does. Therefore stiffness control will not be sensitive to points in the workspace where the Jacobian is singular. Impedance control will require careful trajectory planning or special singular Jacobian error routines to minimize the effects of singularities.

Impedance control uses trajectory information in cartesian coordinates. This is an advantage, because task planning is much easier in cartesian coordinates than in the joint coordinates used for stiffness control. Impedance control also uses a more complete dynamic model of the manipulator. In the impedance control law extra terms are included for the Coriolis and centrifugal forces acting on the robot, and the position, velocity, and force feedback terms are modulated by the manipulator inertia matrix. Because impedance control has a complete dynamic model of the manipulator, in addition to force feedback, it should function well during both the constrained and unconstrained portion of the trajectory. During the free-space portion of the trajectory, $F_{Int} = 0$, and the impedance control law appears as a feedforward dynamics control law. During the constrained portion of the trajectory, the force feedback terms add compliance to the impedance controller allowing it to deal with stiff environments. Stiffness control does not have a comparable mechanism during the unconstrained portion of a trajectory, and will only appear as a PD feedback loop when $F_s = 0$. Although stiffness control allows constraint forces to be specified, impedance control allows the entire dynamic response of the manipulator to be specified through selection of the desired dynamics parameters M, B, and K. Essentially, stiffness control is a lower order form of impedance control, since it involves only the stiffness of the manipulator-environment interaction, not the damping or mass terms. Therefore, impedance control is a much more general form of compliant motion control.

The advantages of impedance control led to its selection as the control law to be used in setting up the compliant motion environment. The selection of *impedance control* should not be taken to mean it is superior to all other possible forms of compliant motion control. Rather, impedance control merely seemed to be the best candidate of those considered for this study.

The selection of impedance control does have one additional benefit. Hogan performed and documented a two degree of freedom impedance control experiment [20] (see Section 2.2.3). In this study the same experiment was repeated, but on a PUMA 560 with a much more complex and non-linear set of dynamics than Hogan's direct drive manipulator. The experiment served as a test of the compliant motion environment and also verified the impedance control law's ability to control a more complex manipulator.

III. Algorithm Development

3.1 Impedance Control

A knowledge of the concepts of impedance and admittance as they apply to robot manipulators is necessary to understand the principals of impedance control. The following theoretical development begins with a presentation of these concepts and motivates their use to generate a model of the desired manipulator dynamic behavior. Given a model of the manipulator's actual dynamics, pole-placement techniques are then used to determine a generalized impedance control law. Next, the assumptions and mathematics used to specialize the control law for this thesis are presented. Finally, the methods used to calculate specific components of the specialized control law are discussed.

The discussion of the development of the general control law (Sections 3.1.1 through 3.1.4) draws heavily from papers by Hogan [17],[18],[19]. The specialization of the control law (Section 3.1.5) is also similar to Hogan's previous research [20]. Section 3.1.6 presents new research unique to the PUMA 560 robot arm.

3.1.1 The Concepts of Impedance and Admittance. The terms impedance and admittance are commonly used in electrical circuit theory. In electrical engineering, impedance is defined as the frequency domain (s-plane) ratio of voltage to current across some electrical network [5, pp.272–274]. For a linear network, admittance is the reciprocal of impedance, that is the ratio of current to voltage. For example, given the simple network of Fig. 3.1 the impedance is found to be

$$\frac{V(s)}{I(s)} = R + Ls + \frac{1}{Cs} \tag{3.1}$$

If more general definitions of impedance and admittance are used, the concepts can be extended to networks of mechanical linkages. Hogan defined the following:

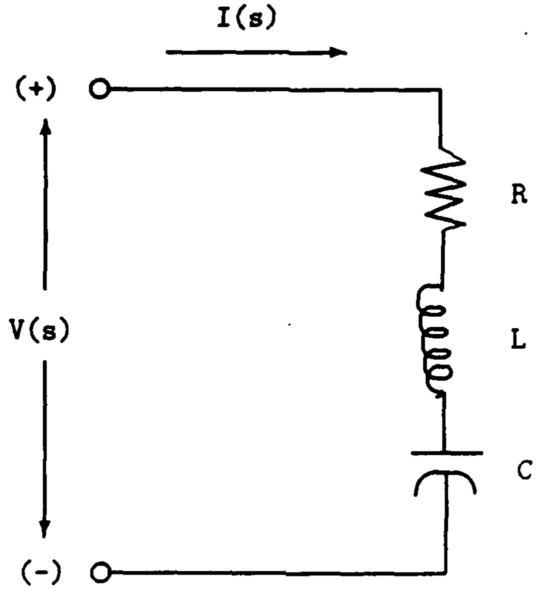


Figure 3.1. Electrical Network [5, p. 274]

Mechanical Impedance — an entity accepting as input flow or motion (e.g. velocity or position) and outputting effort (e.g. force, torque).

Mechanical Admittance — an entity accepting as input effort (e.g. force) and outputting flow or motion (e.g. velocity or position) [17, p. 2].

Position or velocity are the mechanical analogs of charge or current, and force is the mechanical analog of voltage (electromotive force). Returning to electrical systems, it is clear a capacitor with charge q, capacitance C, and characteristic equation

$$q = CV \tag{3.2}$$

can be thought of as an admittance, since it is taking in voltage (effort) and yielding charge (integrated motion). Likewise, for mechanical systems a spring (see Fig. 3.2) acts as an impedance since it accepts displacement (motion) and creates a force (effort). On the other hand, a mass (see Fig. 3.3) is an admittance since given a force input it yields velocity [17, pp. 2–3].

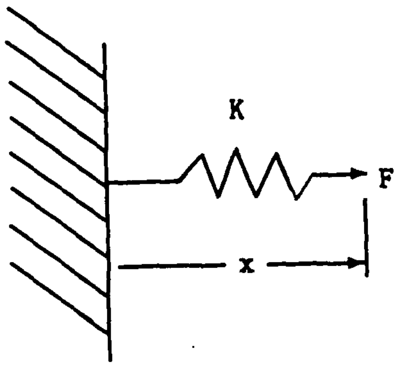


Figure 3.2. Mechanical Spring of Stiffness K, at Position X, Applying Force F

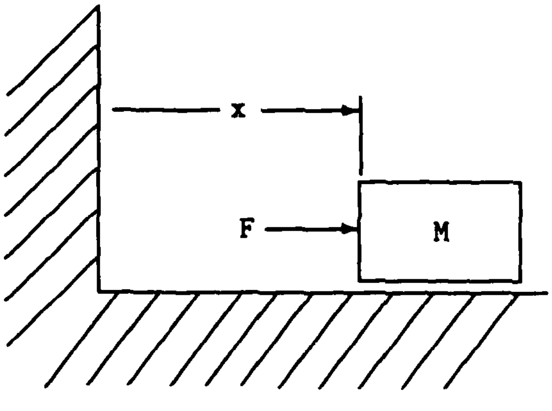


Figure 3.3. Mass M, Acted on by Force F, Being Displaced to Position X

While for linear systems impedance and admittance are reciprocal quantities, Hogan qualitatively proved for non-linear systems, such as robot arms, impedance and admittance are not reciprocals and cannot be interchanged [17, pp. 2–4]. Devices are either fundamentally admittances or impedances, depending on their behavior. For example, consider a mass existing in an environment with non-linear effects such as stiction. If the time history of the forces acting on the object are known, then using

$$v = \frac{1}{m}\int_0^T \sum F dt \tag{3.3}$$

it is possible to determine the velocity of the mass. However, given the time history of velocity it is not possible to determine the time history of force because of the non-differentiable stiction effect. Therefore masses must be treated as admittances, and the manipulator and all objects in the environment are correctly dealt with as admittances [17, p. 4].

For correct theoretical compatibility of non-linear physical systems, Hogan stated impedances must match to admittances as they have corresponding input and output variables. If the manipulator links and objects in the environment must be admittances (since they have mass), then the relationship used by the robot controller must act as an impedance, hence the term impedance control. Since the manipulator and environment are, in general, non-linear that relationship cannot be changed [17, p. 4],[20, p. 1048].

Hogan performed a bond graph analysis with admittances represented as nodes. Using bond graph theory all the impedances acting on the particular admittance can be summed to determine the motion of the response (just like all the forces acting on a mass are summed in Newton's Second law, $\Sigma\vec{F} = m\vec{a}$, to get the equation of motion). The technique can be applied to any admittance, including both inertias in the environment and the inertias of the manipulator links. The universal nature of the technique is key, because it justifies the use of impedance control as a method of driving the manipulator, both when it is in contact with the environment, and when it is not [17, p. 5–6]. Hence, impedance control can be used as a unified approach to the separate control problems of unconstrained position and compliant motion.

Readers can obtain a qualitative feel for the meaning of mechanical impedance and mechanical admittance by considering two results of Hogan's, stated here without proof [19, p. 18]

- For a given trajectory, for maximum power transmission from the manipulator to an object in the environment, the controller's commanded impedance must equal the environment's admittance.
- For minimum deviation from a desired path and simultaneously minimum applied contact force (at the interface between the object and the manipulator), the commanded impedance must be directly proportional to the environment's admittance.

The first result, the concept of impedance matching, is familiar to anyone who has matched a set of stereo speakers to an amplifier. The result is useful in robotics for tasks requiring motion of an object at maximum speed, without regard to contact forces. It indicates the control law should appear exactly as the actual dynamics of the manipulator and payload. In fact, this is done by feedforward dynamic control techniques, although proportional-derivative feedback loops are also typically included to account for errors arising from imperfect modeling of the manipulator dynamics.

The second result is useful for compliant motion control tasks. The qualitative insight to be gained is best stated by Hogan

> If the environment is unyielding (low admittance), the manipulator should accommodate the environment (low impedance); if the environment offers little resistance (high admittance), the manipulator may impose motion upon it (high impedance) [19, p. 18].

For example, consider the robot refueling task. When the nozzle is moving through free-space a high impedance control value may be used (since air has little resistance, i.e. high admittance). However, when the nozzle contacts the aircraft, lower impedance values must be used because the aircraft surface is stiff (low admittance). The end result is the use of force feedback, a measure of the environment's admittance, to adjust the impedance imposed on the system by the controller [20, pp. 1048-1049].

3.1.2 Desired Dynamics. "A general strategy for controlling a manipulator is to control its motion (as in conventional robot control) and in addition give it a 'disturbance response' for deviations from that motion which has the form of an impedance [17, p. 4]". To give the manipulator the correct disturbance response, the first step is to choose the form of the desired impedance for the manipulator. The result will specify the desired dynamic behavior of the system.

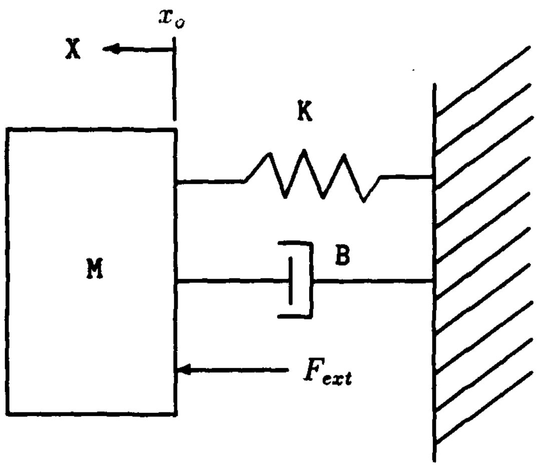
For simplicity, consider the model of the interaction between the manipulator and the environment to be a simple second order mass-spring-damper system (shown in Fig. 3.4). Here the mass, M, is the mass of the manipulator. Let K be the stiffness of a linear spring whose displacement from its normal, relaxed, cartesian position (x_o) is $x_o - x$. The damper has damping coefficient B, and acts with a force linearly proportional to the difference between a nominal cartesian velocity (v_o) and the current velocity v. Then applying Newton's Second law

$$\sum F = M\frac{dv}{dt} \tag{3.4}$$

gives

$$K(x_o - x) + B(v_o - v) + F_{ext} = M\frac{dv}{dt} \tag{3.5}$$

Instead of working with the external force *applied to* the manipulator by the environment (F_{ext}), Hogan chose to use the force *applied by* the manipulator to the



$$F_{spring} = K(x_o - x)$$
$$F_{damper} = B(v_o - v)$$

Figure 3.4. Target Dynamics Model

environment. He called this the interface force (F_{int}) where

$$F_{int} = -F_{ext} \tag{3.6}$$

Substituting Eqn. 3.6 into Eqn. 3.5 and rearranging yields

$$F_{int} = K(x_o - x) + B(v_o - v) - M\frac{dv}{dt} \tag{3.7}$$

The above equation represents the desired dynamics of the manipulator for a single degree of freedom. It can be expanded to a three dimensional, six degree of freedom problem (three degrees of translation and three of rotation) simply by considering F_{int}, x_o, v_o, x, and v to be 6×1 vectors. Then K and B are taken to be 6×6 matrices, and for the simple case of decoupled motion along each of the cartesian axes, K and B are diagonal. The mass term becomes a 6×6 inertia tensor. If the manipulator's *desired* behavior is set up as a rigid body with principal axes along the three cartesian axes, then

$$M = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \tag{3.8}$$

The six degree of freedom desired dynamics are [18, p. 10]

$$\vec{F}_{int} = K(\vec{x}_o - \vec{x}) + B(\vec{v}_o - \vec{v}) - M\frac{d\vec{v}}{dt} \tag{3.9}$$

The values for the M, B, and K control parameter matrices will determine the response of the manipulator as it tracks a commanded trajectory $(\vec{x}_o, \vec{v}_o)$, and/or reacts to a disturbance force (represented here as the interface force, $\vec{F}_{int}$). These parameters can be chosen using conventional control techniques by modeling the system in the Laplace domain (s-plane). Consider the desired dynamics for the scalar case (Eqn. 3.7) written in terms of an error, $e(t)$, where

$$\begin{aligned} e(t) &= x_o - x \\ \dot{e}(t) &= v_o - v \\ \ddot{e}(t) &= \dot{v}_o - \dot{v} \end{aligned} \tag{3.10}$$

For simplicity, the commanded trajectory does not include desired acceleration information, so the desired acceleration is assumed to be zero ($\dot{\vec{v}}_o = 0$). $\dot{v}_o = 0$. Then

$$F_{int} = Ke(t) + B\dot{e}(t) + M\ddot{e}(t) \tag{3.11}$$

and taking the Laplace transform yields the 2nd order system transfer function,

$$\frac{E(s)}{F_{int}(s)} = \frac{1}{s^2 + \frac{B}{M}s + \frac{K}{M}} \tag{3.12}$$

The denominator of Equation 3.12 is the characteristic equation of the desired dynamics. The characteristic equation of a 2nd order system is of the form,

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \tag{3.13}$$

where ω_n is the natural frequency of the desired dynamic system (in radians per second) and ζ is the damping ratio. Comparing Equations 3.13 and 3.12, establishes the following relationships

$$\omega_n^2 = \frac{K}{M} \tag{3.14}$$

and

$$2\zeta\omega_n = \frac{B}{M} \tag{3.15}$$

Given a desired damping ratio, natural frequency, and stiffness for the tool-environment interaction, the necessary values for B and M can be determined. In this thesis, damping ratios of $\zeta \geq 1.0$ are desired to insure the response of the manipulator is critically damped or overdamped, thereby preventing oscillation of the end-effector.

The closed-loop s-plane poles of the desired dynamics are at

$$s = \sigma \pm j\omega_d \tag{3.16}$$

where the damping coefficient, σ, is

$$\sigma = -\zeta\omega_n \tag{3.17}$$

and the damped natural frequency, ω_d, of the system is

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{3.18}$$

The effect of the pole locations on system response is discussed in detail in [7, pp. 75–102].

In this thesis, the desired dynamics are chosen to provide separate, decoupled responses along each degree of freedom. As a result, so the M, B, and K matrices are diagonal. If non-diagonal matrices are used the response can be coupled between several degrees of freedom. In this case, the dynamic response must be analyzed using eigenvector-eigenvalue matrix techniques, although the fundamental principals are the same as the simpler scalar case (these techniques are explained by Kazerooni in [27]).

3.1.3 *Manipulator Dynamics.* The manipulator dynamics are assumed to be of the form given by Hogan [18, p. 10]

$$I(\vec{q})\frac{d\dot{\vec{q}}}{dt} + \vec{C}(\vec{q},\dot{\vec{q}}) + \vec{V}(\dot{\vec{q}}) + \vec{S}(\vec{q}) \;=\; \sum \vec{\tau}_i \tag{3.19}$$

where for a n degree of freedom manipulator

$\vec{q}$ = the vector of joint angular positions ($n \times 1$).

$\dot{\vec{q}}$ = the vector of joint angular velocities ($n \times 1$).

$I(\vec{q})$ = the configuration dependent manipulator inertia tensor ($n \times n$).

$\vec{C}(\vec{q},\dot{\vec{q}})$ = the configuration and velocity dependent vector of Coriolis and centrifugal torques on each joint ($n \times 1$).

$\vec{V}(\dot{\vec{q}})$ = the vector of velocity dependent torques acting on each joint. These are principally friction forces ($n \times 1$).

$\vec{S}(\vec{q})$ = the vector of position dependent torques acting on each joint. The most significant component is typically gravity ($n \times 1$).

$\sum \vec{\tau}_i$ = the sum of the torques applied at the links. If the manipulator is in free-space then $\sum \vec{\tau}_i$ only includes the actuator torques ($\vec{\tau}_{act}$). If the manipulator is in contact with the environment, then $\sum \vec{\tau}_i = \vec{\tau}_{act} + \vec{\tau}_{int}$ where $\vec{\tau}_{int}$ is the vector of interface torques imposed on each link by contact with the environment.

The equation for the manipulator dynamics can include all the dynamic effects typically incorporated into a manipulator model. It is equivalent to the closed chain Lagrange-Euler dynamic model used by Walker and Orin [57, p. 207], and functionally equivalent to the closed chain dynamics presented by Potkonjak and Vukobratovic [46, p. 322].

The dynamic model does not include terms to account for gear train dynamics (except for friction) and actuator dynamics (except for the motor inertia incorporated into $I(\vec{q})$). As a result, the minor, non-linear effects of gear backlash and motor cogging are not modeled. The links are assumed to be inflexible, and any high order vibrational dynamics of the links are not considered significant enough to be modeled (a good assumption for a very stiff structure like the PUMA).

With a reasonably accurate model of the manipulator established, it is possible to proceed with the control law derivation.

3.1.4 *Control Law.* Pole placement techniques are used to derive the control law by comparing the model of the desired dynamics (Eqn. 3.9) to the model of the actual dynamics (Eqn. 3.19). It is possible to algebraically solve for $\vec{\tau}_{act}$ as a function of the commanded cartesian position and velocity ($\vec{x}_o$, $\vec{v}_o$), the interface forces ($\vec{F}_{int}$), and the desired dynamic parameters M, B and K (see Appendix A). The result for an n degree of freedom arm is

$$\begin{aligned}
\vec{\tau}_{act} = {} & I(\vec{q})J(\vec{q})^{-1}M^{-1}K\left[\vec{x}_o - \vec{L}(\vec{q})\right] + \vec{S}(\vec{q}) \\
& + I(\vec{q})J(\vec{q})^{-1}M^{-1}B\left[\vec{v}_o - J(\vec{q})\dot{\vec{q}}\right] + \vec{V}(\dot{\vec{q}}) \\
& - \left[J(\vec{q})^T + I(\vec{q})J(\vec{q})^{-1}M^{-1}\right]\vec{F}_{int} \\
& - I(\vec{q})J(\vec{q})^{-1}\vec{G}(\vec{q},\dot{\vec{q}}) + \vec{C}(\vec{q},\dot{\vec{q}})
\end{aligned} \tag{3.20}$$

where, in addition to the components already defined,

$\vec{L}(\vec{q})$ = the manipulator forward kinematics function. A vector function yielding cartesian position and orientation as a function of the joint angles ($n \times 1$).

$J(\vec{q})$ = the manipulator Jacobian $J(\vec{q}) = \frac{d\vec{L}(\vec{q})}{d\vec{q}}$ a matrix ($n \times n$)

$\vec{G}(\vec{q},\dot{\vec{q}})$ = the vector of Coriolis and centrifugal torques resulting from differentiation of the Jacobian (see Appendix A).

Equation 3.20 has been arranged so the first row contains the position dependent terms, the second row the velocity dependent terms, the third row the force feedback terms, and the last row has the inertial coupling and Coriolis and centrifugal terms. Equation 3.20 is well described by Hogan as "a non-linear feedback law relating actuator torques to observations of actuator position, velocity, and interface force," and it "expresses the required behavior to be provided by the controller... a non-linear impedance in actuator coordinates [18, p. 10]."

Hogan claims the full control law requires about the same number of computations as the exact, recursive Lagrange-Euler or Newton-Euler open chain dynamic equations [18, p. 10]. For a six link serial manipulator using the Lagrange-Euler technique, Hollerbach reported a requirement for 2195 to 4388 multiplications and 1719 to 3586 additions, depending on the order of the homogeneous transformation matrices used in the computation [21, p. 732]. In this study the number of computations were reduced by making assumptions about the task, thereby allowing significant simplification of Eqn. 3.20. These assumptions and the resulting simplified control law are discussed in the next section.

3.1.5 *Simplified Control Law* The full control law presented in Eqn. 3.20 is computationally involved. For less demanding compliant motion tasks it may represent an unnecessarily high degree of modeling accuracy. This degree of accuracy may be difficult to implement in a real-time digital control microprocessor. In this study the complexity of the control law was reduced by making a few limiting assumptions about the nature of the task.

By assuming low joint velocities, the Coriolis and centrifugal terms ($\vec{G}(\vec{q},\dot{\vec{q}})$ and $\vec{C}(\vec{q},\dot{\vec{q}})$) can be assumed to be negligible. This is a reasonable restriction for most compliant motion tasks since high speed motion is not usually required, or in some cases even desirable.[1] Dropping the $\vec{G}$ and $\vec{C}$ functions from the control law significantly reduces the number of calculations required. For example, even with symbolic reduction of the Lagrange-Euler equations, determining $\vec{C}(\vec{q}, \dot{\vec{q}})$ for a six link robot arm requires 215 multiplies and 142 adds (50 % of the total operations required) [35].

If the velocities are assumed small, the commanded cartesian velocity can be approximated as $\vec{v}_o = 0$. Making this assumption does not significantly reduce the number of computations required, but it does simplify the handling of trajectory information. Trajectories no longer have to be created with a look-up table of velocity versus time, corresponding to a similar table of position versus time. The table of velocity values does not have to be loaded into the limited memory of the control computer, and the real-time control algorithm does not have to interpolate velocity values in between trajectory set-points. Lastly, the elimination of $\vec{v}_o$ allows the $I(\vec{q})J(\vec{q})^{-1}M^{-1}B$ and $-J(\vec{q})$ matrices of the control law damping term to be combined into a single term, $-I(\vec{q})J(\vec{q})^{-1}M^{-1}BJ(\vec{q})$. The combined term can be calculated asynchronously in a separate parallel processor to reduce the computational load in the primary control processor. Unfortunately, lack of desired velocity information will degrade the ability of the manipulator to track fast trajectories. For accurate well behaved tracking of trajectories, at moderate to high speeds, provisions should be made for including $\vec{v}_o$ information in the control algorithm.

Further reduction in the number of control law terms cannot be theoretically justified when the law is being applied to a vertically articulated, gear driven robot like the PUMA 560. The large gravity component in the PUMA dynamics prevents the elimination of the $\vec{S}(\vec{q})$ term. For example, with no load, the gravitational torque on link 2 varies from -0.72 N-m to 62.5 N-m. Also, $\vec{V}(\dot{\vec{q}})$ cannot be dropped because of the substantial friction present in the gear train [36, pp. 6–10], [32, pp. 20–21].

---

[1] However, elimination of these terms will decrease the accuracy of the controller at high speed, and will limit the impedance control law's ability to serve as a unified control law for use in both compliant motion tasks and freespace trajectory tracking tasks.

Based on these assumptions, the control law implemented in this study is

$$\begin{aligned} \vec{\tau}_{act} &= I(\vec{q})J(\vec{q})^{-1}M^{-1}K\left[\vec{x}_o - \vec{L}(\vec{q})\right] + \vec{S}(\vec{q}) - I(\vec{q})J(\vec{q})^{-1}M^{-1}BJ(\vec{q})\dot{\vec{q}} + \vec{V}(\dot{\vec{q}}) \\ &\quad - \left[J(\vec{q})^T + I(\vec{q})J(\vec{q})^{-1}M^{-1}\right]\vec{F}_{int} \qquad (3.21) \end{aligned}$$

The simplified demonstration of impedance control performed here only requires links 2 and 3 to be governed by the impedance control law. Links 1, 4, 5, and 6 are assumed to be fixed at the position $q = 0$. Therefore compliant motion is restricted to the world coordinate xz plane. This dramatically reduces the number of computations required. The reduction occurs because the order of the matrices and vectors involved is only $n = 2$ instead of $n = 6$. A further reduction occurs because the equations required to calculate elements of $I(\vec{q})$, $J(\vec{q})$, $\vec{L}(\vec{q})$, and $\vec{S}(\vec{q})$ are greatly simplified. The choice of $q = 0$ for the inactive joints is not significant in itself. Any fixed value of q for these four joints would be sufficient, although the use of $q \neq 0$ would alter the constant values used in symbolically determining $I(\vec{q})$, $J(\vec{q})$, $\vec{L}(\vec{q})$, and $\vec{S}(\vec{q})$. Control of joints 1, 4, 5, and 6 is performed using a simple proportional-derivative feedback loop as discussed in Section 3.3.

The simplified control law, Eqn. 3.21, is similar to the control law used by Hogan for an experimental demonstration of impedance control [20, p. 1050]. He included a full calculation of the Coriolis and centrifugal, ($\vec{G}(\vec{q},\dot{\vec{q}})$ and $\vec{C}(\vec{q},\dot{\vec{q}})$), torques. He implemented the control law on a specially designed, direct-drive, two link robot arm. The arm had very low joint friction values, so $\vec{V}(\dot{\vec{q}})$ was not included in the control law. The arm was also horizontally articulated, so gravity compensation ($\vec{S}(\vec{q})$) was not necessary.

3.1.6 *Development of Control Law Components.* Equation 3.21 provides the control law for joints 2 and 3. To calculate the actuator torques on these links it is first necessary to determine values for the inertia tensor, Jacobian, gravity torque vector, forward kinematics, and friction torque vector. Development of symbolic equations for $I(\vec{q})$, $J(\vec{q})$, $J(\vec{q})^{-1}$, $\vec{S}(\vec{q})$, and $\vec{L}(\vec{q})$ is described below and in Sections 3.1.6.1 through 3.1.6.4. An existing subroutine is used to calculate the viscous and Coulomb friction forces comprising $\vec{V}(\dot{\vec{q}})$ [33, p. 10]. The calculation of $\vec{F}_{int}$ based on forces and moments measured by the force sensor is discussed in Section 3.2.

In general the quantities $I(\vec{q})$, $J(\vec{q})$, $J(\vec{q})^{-1}$, $\vec{S}(\vec{q})$, and $\vec{L}(\vec{q})$ are symbolically determined for the case where joints 1, 4, 5, and 6 were positioned at $q = 0$. Substituting the Denavit-Hartenberg parameters describing the PUMA 560 (see Fig. 4.2) into the symbolic relationships creates the equations implemented in the digital processor. Parameters relating to the tool or payload dimensions and mass are left in symbolic form to allow for greater user flexibility. The control law equations are also left as functions of the desired dynamics parameters, M, B, and K. The resulting equations are relatively compact and straight-forward to code.

3.1.6.1 *Forward Kinematics.*

The forward kinematics were developed symbolically using a MACSYMA robotics CAD program intended for symbolic construction of a manipulator's kinematic and dynamic equations. The program was developed by Leu and Hemati [41], and implemented at AFIT by S. Parker in 1988.

The forward kinematics consist of the ${}^0T_6$ 4×4 homogeneous transformation matrix, where

$$ {}^0T_6 = \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{{}^0R_6} & \vec{p} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \begin{bmatrix} \vec{n} & \vec{s} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.22) $$

For determining the cartesian position based on joint angles, the forward kinematics function ($L(\vec{q})$) need only compute the tool position vector ($\vec{p}$). Because only joints 2 and 3 are controlled by the impedance controller, and all the other joints are fixed at $q = 0$, motion only occurs in the world xz plane. Therefore, only p_x and p_z are calculated since p_y is fixed.

Using MACSYMA the forward kinematics were found to be

$$\begin{aligned} p_x &= d_{64}\sin(q_2+q_3) + a_3\cos(q_2+q_3) + a_2\cos(q_2) \\ p_y &= d_2 \\ p_z &= d_{64}\cos(q_2+q_3) - a_3\sin(q_2+q_3) - a_2\cos(q_2) \end{aligned} \tag{3.23}$$

where a_i, and d_i are the Denavit-Hartenberg parameters for the ith link as given in Fig. 4.2. The parameter d_{64} is uniquely defined for this study as

$$d_{64} = d_6 + d_4 + L_{tool} \tag{3.24}$$

The variable L_{tool} is the distance along the joint 6 $\hat{z}$-axis from the joint 6 tool mounting flange to the tip of the tool. Equations 3.23 agree with those published in [16, p. 45] where $q = 0$ for joints 1, 4, 5, and 6.

Knowledge of ${}^0R_6$ is not directly necessary for computing components of Eqn. 3.21, but it is used in obtaining $\vec{F}_{int}$ in the world coordinate frame (see Section 3.2.5). For purposes of finding the interface force, $\vec{q}$ can not be assumed to be zero for joints 1, 4, 5, and 6. A complete, accurate set of joint angles is required to insure the orientation of the tool is correctly computed. As a result the symbolic equations for ${}^0R_6$ are a function of all six joint angles and are used exactly as they appear in the literature [16, p. 45].

3.1.6.2 *Jacobian.* The manipulator Jacobian relates values described in joint space coordinates to values expressed in cartesian coordinates. The Jacobian is commonly defined by [43, pp. 10–11], [16, p. 544] as

$$\vec{v} = J\dot{\vec{q}} \tag{3.25}$$

Given a forward kinematics function, $\vec{L}(\vec{q})$, calculating the cartesian position vector as a function of $\vec{q}$ then

$$\vec{x} = \vec{L}(\vec{q}) \tag{3.26}$$

and the Jacobian is found from differentiation

$$\vec{v} = \dot{\vec{x}} = \frac{d\vec{L}(\vec{q})}{d\vec{q}}\dot{\vec{q}} \tag{3.27}$$

as[2]

$$J = \frac{d\vec{L}(\vec{q})}{d\vec{q}} \tag{3.28}$$

For a n degree of freedom manipulator with m links the Jacobian is $n \times m$.

Since only x and z position, velocity, or force values are being considered in the control law, and only links 2 and 3 are being controlled, the Jacobian is 2×2. Of the general 6×6 Jacobian, only the J_{12}, J_{13}, J_{32}, and J_{33} elements are used. From the definition of $\vec{L}(\vec{q})$ and Eqn. 3.28 the 2×2 Jacobian is

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} \\ \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} \end{bmatrix} \tag{3.29}$$

Using MACSYMA the Jacobian elements were found to be

$$\begin{aligned} J_{11} &= d_{64}\cos(q_2+q_3) - a_3\sin(q_2+q_3) - a_2\sin(q_2) \\ J_{12} &= d_{64}\cos(q_2+q_3) - a_3\sin(q_2+q_3) \\ J_{21} &= -d_{64}\sin(q_2+q_3) - a_3\cos(q_2+q_3) - a_2\cos(q_2) \\ J_{22} &= -d_{64}\sin(q_2+q_3) - a_3\cos(q_2+q_3) \end{aligned} \tag{3.30}$$

MACSYMA's powerful symbolic manipulation capabilities were used to find the inverse Jacobian (J^{-1}) as

$$\begin{aligned} J_{11}^{-1} &= \frac{1}{\Delta}\left[d_{64}\sin(q_2+q_3) + a_3\cos(q_2+q_3)\right] \\ J_{12}^{-1} &= \frac{1}{\Delta}\left[-d_{64}\cos(q_2+q_3) + a_3\sin(q_2+q_3)\right] \\ J_{21}^{-1} &= \frac{1}{\Delta}\left[d_{64}\sin(q_2+q_3) + a_3\cos(q_2+q_3) + a_2\cos(q_2)\right] \\ J_{22}^{-1} &= \frac{1}{\Delta}\left[-d_{64}\cos(q_2+q_3) + a_3\sin(q_2+q_3) + a_2\sin(q_2)\right] \end{aligned} \tag{3.31}$$

where $\Delta = \det(J) = a_2\left[a_3\sin(q_3) - d_{64}\cos(q_3)\right]$

---

[2] This definition of the Jacobian is invariant for the xyz position elements, but for orientation angles it is dependent on the choice of angles [43, pp. 10–11].

3.1.6.3 Manipulator Inertia Tensor. The manipulator inertia tensor represents the effective and coupling inertias of all the manipulator links as seen by the actuators. More specifically, the I_{ii} element of the inertia tensor is the inertia seen by the actuator at the ith joint as it accelerates the combined inertia of the ith link and all the other links placed in motion by the movement of the ith link (i.e. if link 3 is attached to link 2, then moving link 2 causes link 3 to move). "The I_{ij} element of the inertia tensor is related to the reaction torque ...induced by the acceleration of joint j and acting on joint i, or vice versa [16, p. 96]". The reflected inertia of the ith motor is included with the I_{ii} term, since for a gear driven robot the motor's own inertia appears as a significant portion of the overall load. As with a rigid body, the manipulator inertia tensor is symmetric.

For an n degree of freedom manipulator the inertia tensor is $n \times n$. This impedance controller only deals with links 2 and 3, so the inertia is a just a 2×2 subset of the full 6×6 six degree of freedom inertia tensor. In this case, $I(\vec{q})$ consists of the I_{22}, I_{23}, I_{32}, and I_{33} elements of the 6×6 inertia tensor. For simplicity these values are renamed I_{11}, I_{12}, I_{21}, and I_{22}.

Tarn and Bejczy provided symbolic equations for the effective and coupling inertias of links 2 and 3, assuming joints 4, 5, and 6 (the wrist joints) are at $q = 0$ [52, pp. 12–14]. These were loaded into a MACSYMA batch file along with the values of the PUMA 560 Denavit-Hartenberg parameters, the link masses, and the radii of gyration of the individual links[3]. MACSYMA was then used to develop the symbolic equations for $I(\vec{q})$ as function of q_2, q_3, and the payload parameters. The resulting equations are

$$\begin{aligned} I_{11} &= \tilde{m}_3 \left[0.8636\,(\overline{z}_3 \sin(q_3) + (\overline{x}_3 - 0.0191)\cos(q_3)) + k_{3y}^2 - 0.0382\overline{x}_3 + 0.18682\right] \\ &\quad + 3.5654 \\ I_{12} &= \tilde{m}_3 \left[0.4318\,(\overline{z}_3 \sin(q_3) + (\overline{x}_3 - .0191)\cos(q_3)) + k_{3y}^2 - 0.0382\overline{x}_3 + 0.00036\right] \end{aligned}$$

---

[3] The MACSYMA batch file routine was developed at AFIT by L. Tellman in 1988.

$$
\begin{aligned}
I_{21} &= I_{12} \\
I_{22} &= \tilde{m}_3 \left[k_{3y}^2 - 0.0382\bar{x}_3 + 0.00036 \right] + 0.5827
\end{aligned} \tag{3.32}
$$

The terms $\tilde{m}_3$, $\bar{x}_3$, $\bar{z}_3$, and k_{3y} are inertial parameters of link 3 affected by the user determined payload mass. The total mass of the third link, wrist, and payload is $\tilde{m}_3$.

$$
\tilde{m}_3 = m_3 + m_{wrist} + m_{payload} \tag{3.33}
$$

Tarn gives the sum $m_3 + m_{wrist}$ as 6.97 kilograms for the PUMA [52, p. 18]. For this study $m_{payload}$ consisted of the force sensor and tool masses (m_{Tool}).

The location of the third link center of gravity, in the third link coordinate frame, is given by $\bar{x}_3$ and $\bar{z}_3$. These values are a function of the mass and center of gravity locations of the payload and wrist. The payload is assumed to be symmetric, with a center of gravity along the $\hat{z}_6$ coordinate axis. Since $q_6 = 0$, the $\hat{z}_6$ and $\hat{z}_3$ axes are aligned and the combined link 3, wrist, and payload center of gravity is not affected in the $\hat{y}_3$ direction by variation in payload mass. Therefore, $\bar{y}_3$ is constant and has been included numerically in the inertia equations. Using the definition of the center of mass, and the combined link 3 plus wrist center of mass given by [52, p. 18], $\bar{x}_3$ and $\bar{z}_3$ can be found to be

$$
\begin{aligned}
\bar{x}_3 &= \frac{\bar{\bar{x}}_3}{\tilde{m}_3} \\
\bar{z}_3 &= \frac{\bar{\bar{z}}_3 + \bar{r}_6}{\tilde{m}_3}
\end{aligned} \tag{3.34}
$$

where $\bar{\bar{x}}_3$ and $\bar{\bar{z}}_3$ are the unloaded location of the link 3 plus wrist center of mass (0.0136 meter and 0.1522 meter respectively) [52, p. 18].

The last payload parameter, k_{3y}, is the radius of gyration about the $\hat{y}_3$ axis for the third link, wrist, and payload combination. Although this is affected by the payload, the effect has been shown to be insignificant, particularly for small payload mass [38]. As a result, k_{3y} was kept constant at the value given by Tarn for the unloaded third link and wrist (0.28035 meter) [52, p. 18].

3.1.6.4 *Gravity Torque Vector* The gravity torque vector represents the torque at the ith joint, due to the force of gravity acting on the combined center of mass for all the links between the ith joint and the payload (including the payload mass). The gravitational acceleration g equals 9.8062 m/s^2. Tarn and Bejczy provide the symbolic equations for the gravitational torque on links 2 and 3 in [52, p. 17]. These were already developed for the wrist joints fixed at zero, so $\vec{S}(\vec{q})$ is

$$\begin{aligned} S_1 &= S_2 - g a_2 \tilde{m}_3 \cos(q_2) + g m_2 \left[\bar{y}_2 \sin(q_2) - (\bar{x}_2 + a_2) \cos(q_2)\right] \\ S_2 &= -g \tilde{m}_3 \left[(\bar{x}_3 + a_3) \cos(q_2 + q_3) - \bar{z}_3 \sin(q_2 + q_3)\right] \end{aligned} \tag{3.35}$$

3.2 *Force Sensing.*

The impedance control law requires knowledge of the forces and moments applied by the manipulator at the interface with the environment ($\vec{F}_{int}$). The interface force cannot be measured directly with a wrist mounted force sensor, since the sensor measurements will include tool weight and invariably some sensor bias errors. Also, a wrist mounted force sensor only provides force measurements in the sensor frame. However, it is desirable to have interface force expressed in world coordinates because this fixed frame is typically the easiest to use for task planning. Therefore, to get interface force the outputs of the force sensor must be scaled, calibrated, and then transformed to a tool frame located at the point of contact with the environment; from the tool frame the force information must be transformed to the world coordinate frame where it can be compensated for the force of gravity on the tool and sensor. Only after all these manipulations is the force sensor data ready to be used as $\vec{F}_{int}$ in the control law.

3.2.1 *Force Data Scaling.* The output of the JR3 force sensor is a 6 × 1 vector of measured forces and moments ($\vec{F}_m$). The JR3 sensor provides this information over the parallel (Direct Memory Access) interface as a signed number of

A/D counts. The sensor uses a 12 bit A/D converter with a bipolar range of 2048 counts (i.e. $\frac{1}{2}2^{12}$). Floating point values of measured force are then obtained with

$$\vec{F}_m = \frac{F_{scale}}{2048}\vec{F}_{m,counts} \tag{3.36}$$

where F_{scale} is the diagonal matrix of full scale values for each of the six measured forces and moments and $\vec{F}_{m,counts}$ is the same as $\vec{F}_m$ but is expressed in A/D counts.. Nominally, the elements of F_{scale} are set equal to the force or moment load capacity of the sensor along the associated axis (see Section 4.2). As elements of F_{scale} increase the resolution of the corresponding force measurements decrease, although greater values of force can be measured. According to JR3, increasing F_{scale} beyond the rated load capacity of the sensor allows overload conditions to be measured [24, p. 3-3].

For any individual test F_{scale} is constant so Eqn. 3.36 can be simplified to

$$\vec{F}_m = F_{scl}\vec{F}_{m,counts} \tag{3.37}$$

where $F_{scl} = \frac{F_{scale}}{2048}$ is a diagonal matrix of constants.

3.2.2 Calibration and External Force in the Sensor Frame. The output of the sensor is postulated to contain three types of errors,

- Known Constant Errors — such as data offsets included by the sensor.
- Unknown Constant Errors — such as forces created by bolting the tool to the sensor, and errors due to sensor gain drift.
- Random Noise — if this is assumed to be zero mean, stationary, uncorrelated, and Gaussian it can be ignored for a sufficiently large number of measurements. Since the forces are sampled at a high rate, a typical compliant motion trajectory should contain a sufficient number of measurements for this to be true.

The constant errors can be measured and summed to form a 6 × 1 vector of calibration constants, $\vec{F}_c$. Because these errors occur in the force sensor measurements *in the sensor frame*, they are most easily accounted for before the measurements are transformed to any other frame. Then the total external force on the tool ($\vec{F}_{totext}$) as measured in the sensor frame is

$$\vec{F}_{totext} = \vec{F}_m - \vec{F}_c = F_{scl}\vec{F}_{m,counts} - \vec{F}_c \qquad (3.38)$$

3.2.3 *Limit Checking.* A force sensor can be damaged by application of excessive force to the force transducer. Forces above the ultimate mechanical strength of the transducer will result in fracture of the internal force sensing element. Forces above the yield strength of the transducer can cause plastic deformation of the sensing element, permanently altering its elasticity. When this occurs the sensor must be recalibrated to account for the altered relationship between the measured strains and the applied forces or moments.

To prevent damage to the sensor, the forces on the sensor, in the sensor frame, are compared to the user defined maximum allowable force ($\vec{F}_{max}$). If $\vec{F}_{totext} > \vec{F}_{max}$ is true then an error condition is generated, causing the arm motion to stop, and alerting the operator. The maximum allowable force is easily defined as a function of the full scale force,

$$\vec{F}_{max} = F_{scale} \times (LIMIT) \qquad (3.39)$$

where $LIMIT$ is a 6 × 1 vector of limit factors. Note, the choice of F_{scale} does not place an upper bound on $\vec{F}_{max}$ since $LIMIT$ values can be greater than one.

3.2.4 *Transformation to Tool Frame.* For compliant motion tasks involving a tool in contact with the environment, the force and moments applied by the manipulator are best measured in a coordinate frame with an origin at the point of contact. Using this frame allows any moments applied by contact with the environment to be specifically determined. Using a coordinate frame remote from the point of contact results in moments being measured which are not truly the result of contact with the environment. Instead, the measured moments include components generated by forces acting at a distance from the point about which moments are measured (see Fig. 3.5).[4]

Because the wrist mounted force sensor must be placed between the tool and the manipulator, its coordinate frame cannot be coincident with the point of contact. However, using a corrected version of the technique presented in [50], it is possible to translate the sensed forces and moments to a coordinate frame at the working end of the tool (the tool or nsa frame), and eliminate any "artificial" moments caused by contact forces acting on the tool at a distance from the sensor coordinate frame.

Assuming the sensor and tool frames are parallel, but displaced by some distance $|\vec{r}|$, then from Fig. 3.5

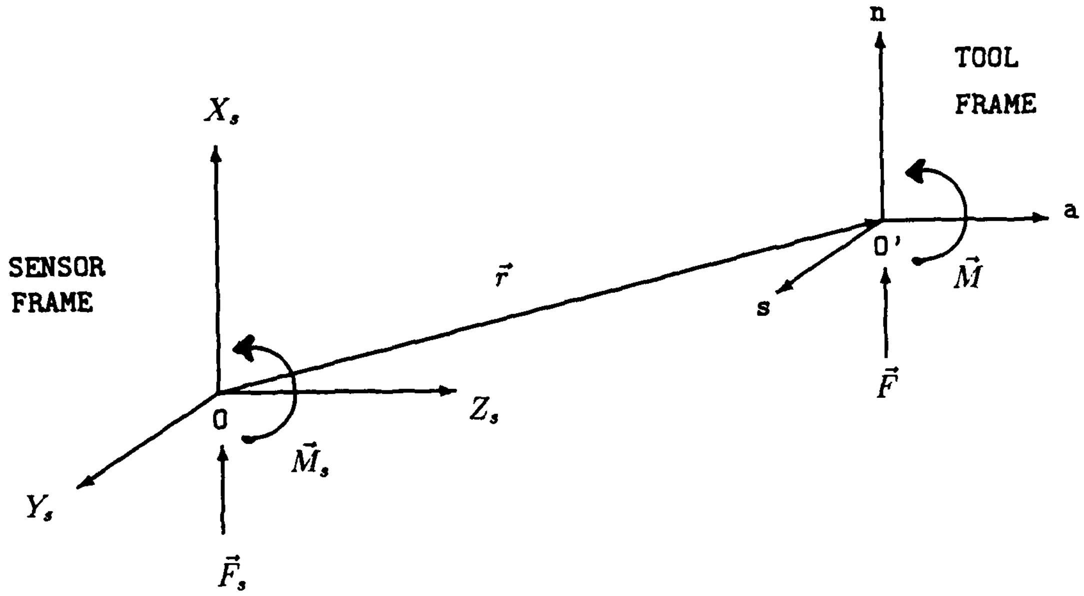
$$\vec{M} = \vec{M}_s - \left(\vec{r} \times \vec{F}\right) \tag{3.40}$$

since $\vec{r}$ is known for a rigid tool and $\vec{F}$ is measured, it is possible to calculate the moments at the contact point. The sensed forces and moments can be translated to the tool frame using the relationship

$$-\left(\vec{r} \times \vec{F}\right) = \begin{bmatrix} 0 & r_z & -r_y \\ -r_z & 0 & r_x \\ r_y & -r_x & 0 \end{bmatrix} \vec{F} \tag{3.41}$$

---

[4] True contact moments are defined to be those generated by a force couple applied about an axis of rotation. Examples are the moment exerted on a screwdriver by a screw, or the moment imparted to a beam by a rotational spring attached to one end.


n
TOOL
FRAME
X_s
a
O'
SENSOR
FRAME
$\vec{r}$
s
$\vec{M}$
$\vec{F}$
O
Z_s
$\vec{M}_s$
Y_s
$\vec{F}_s$

O' is the contact point

$\vec{F} =$ Contact Force

$\vec{M} =$ "True" Contact Moment

$\vec{F}_s =$ Sensed Force

$\vec{M}_s =$ Sensed Moment

Since coordinate frames are parallel, $\vec{F}_s = \vec{F}$

But frames are not coincident $(r \neq 0)$, so

$$\vec{M}_s = \vec{M} + (\vec{r} \times \vec{F})$$

and the sensed moment is not the same as the "true" contact moment.

Figure 3.5. True Moments vs. Sensed Moments

to get the translation equation

$$\begin{bmatrix} F_{totext,x} \\ F_{totext,y} \\ F_{totext,z} \\ M_{totext,x} \\ M_{totext,y} \\ M_{totext,z} \end{bmatrix}_{nsa} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & r_z & -r_y & 1 & 0 & 0 \\ -r_z & 0 & r_x & 0 & 1 & 0 \\ r_y & -r_x & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{totext,x} \\ F_{totext,y} \\ F_{totext,z} \\ M_{totext,x} \\ M_{totext,y} \\ M_{totext,z} \end{bmatrix}_{Sensor} \tag{3.42}$$

For the nozzle simulator used in this study the tool frame is parallel to the sensor frame and the sensor $\hat{z}$ axis is coincident with the $\hat{a}$ axis (see Fig. 3.6). Since the diameter of the nozzle is much less than the tool length, it is ignored, thus contact is assumed to occur along the tool centerline ($\hat{a}$ axis) at the end of the tool. According to JR3, the sensor coordinate frame is located at the center of the transducer [24, p. 4-24]. Labelling the distance between the sensor and the tool frames as the "sensed tool length" (L_{TS}) then

$$L_{TS} = \frac{(Transducer\ Thickness)}{2} + L_{Tool} \tag{3.43}$$

and

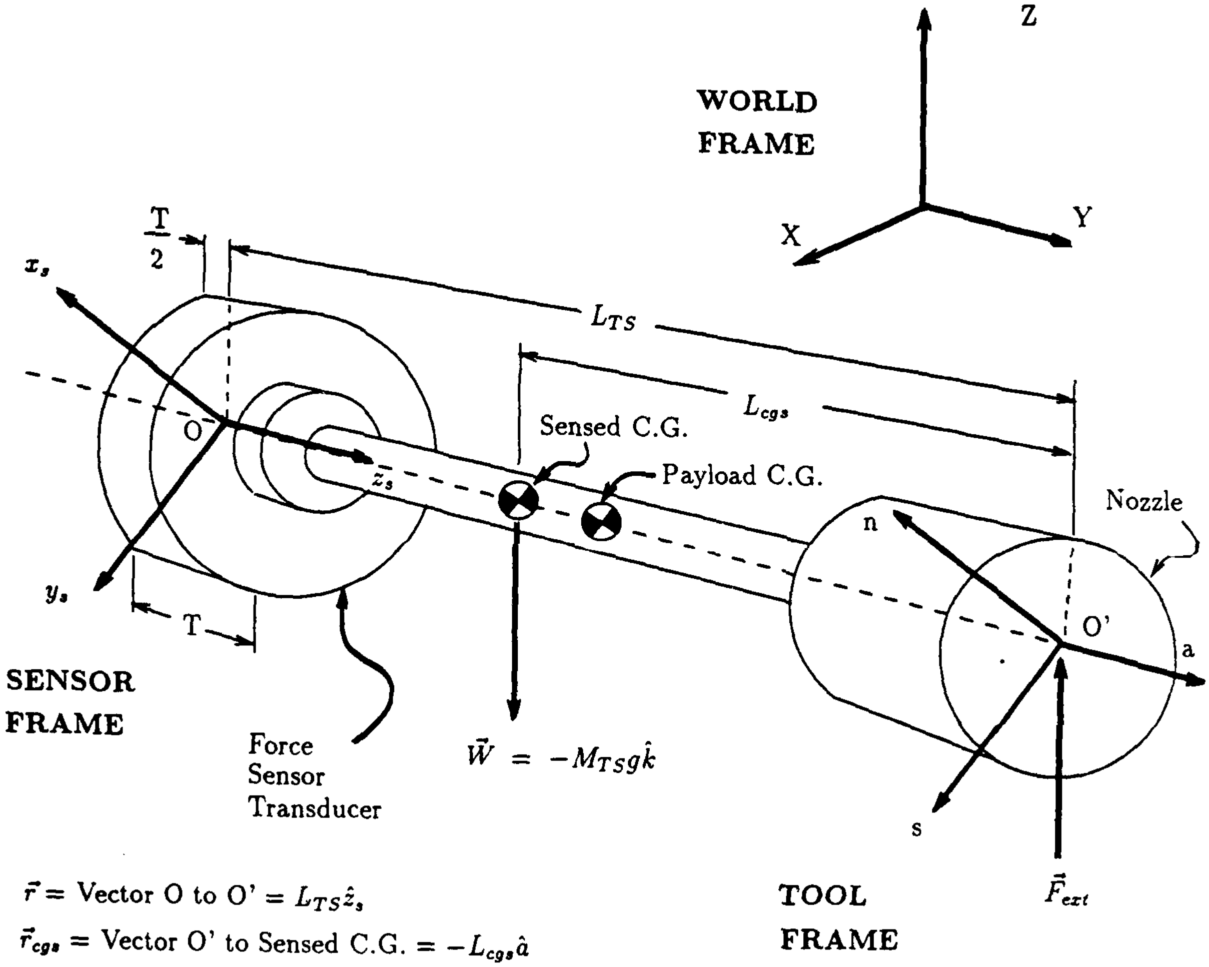
$$\vec{r} = L_{TS}\hat{z}_s \tag{3.44}$$

Therefore, the only effect of the translation between frames is on the moments about the x_s and y_s axes

$$[M_{totext,x}]_{nsa} = [M_{totext,x}]_{Sensor} + L_{TS}F_{totext,y} \tag{3.45}$$

$$[M_{totext,y}]_{nsa} = [M_{totext,y}]_{Sensor} - L_{TS}F_{totext,x} \tag{3.46}$$

3.2.5 Transformation to World Coordinates and Gravity Correction. Equation 3.42 yields the calibrated external force in tool coordinates. Unfortunately, knowledge of forces and moments in the tool coordinates is not particularly useful


Z
WORLD FRAME
X
Y
$\frac{T}{2}$
x_s
L_{TS}
L_{cgs}
O
Sensed C.G.
Payload C.G.
Nozzle
$\hat{z}_s$
n
O'
a
y_s
T
SENSOR FRAME
Force Sensor Transducer
$\vec{W} = -M_{TS} g \hat{k}$
s
TOOL FRAME
$\vec{F}_{ext}$

$\vec{r}$ = Vector O to O' = $L_{TS}\hat{z}_s$

$\vec{r}_{cgs}$ = Vector O' to Sensed C.G. = $-L_{cgs}\hat{a}$

Figure 3.6. Tool and Sensor Diagram

for a user who has surfaces and objects of the environment defined in world coordinates. Since world coordinates are fixed, they provide a much more intuitive basis for task planning. If cartesian trajectories are described in world coordinates, then forces must also be described in the same coordinates for consistency within the control law. As a result, a transformation of $\vec{F}_{totext}$ between the tool coordinates and world coordinates is needed.

From the robot kinematics, a vector $\vec{A}_i$ expressed in the ith frame can be transformed to an alternative frame (j) using the rotation matrix ${}^jR_i$

$$\vec{A}_j = {}^jR_i\vec{A}_i \tag{3.47}$$

Let i and j designate the coordinate frames assigned to the ith and jth links (see Fig. 4.2) of the PUMA arm. Designating the world coordinates as j=0 and the sixth link coordinate frame as i=6, the transformation between the two is ${}^0R_6$. It is possible to mount the force sensor and the tool with their coordinate frames parallel to the joint 6 coordinate frame, then for purposes of rotating vectors from one frame to another,

$${}^0R_6 = {}^0R_{nsa} \tag{3.48}$$

The rotation matrix ${}^0R_6$ is contained within the upper left 3×3 block of the homogenous transformation matrix for the complete forward kinematics

$${}^0T_6 = \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{{}^0R_6} & \vec{p} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \begin{bmatrix} \vec{n} & \vec{s} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.49}$$

and

$${}^0R_6 = \begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix} \tag{3.50}$$

Using Eqn. 3.47 to rotate the vector of external forces and moments from the tool to world frames gives

$$[F_{totext}]_0 = {}^0R_6\,[F_{totext}]_{nsa} \tag{3.51}$$

The total external force acting on the tool consists of forces and moments applied at a point by objects in the environment ($\vec{F}_{ext}$), and body forces acting on the entire tool. At low velocities and accelerations the forces due to tool inertia and the Coriolis and centrifugal accelerations are small. Therefore the only significant body force acting on the tool is gravity ($\vec{F}_g$) so

$$\vec{F}_{totext} = \vec{F}_{ext} + \vec{F}_g \tag{3.52}$$

The interface force is needed in the control law, and from Section 3.1.2

$$\vec{F}_{int} = -\vec{F}_{ext} \tag{3.53}$$

so combining with Eqns. 3.51 and 3.52 results in

$$[F_{int}]_0 = -\,{}^0R_6\,[F_{totext}]_{nsa} + [F_g]_0 \tag{3.54}$$

Examining Fig. 3.6, in world coordinates (represented by the unit vectors $\hat{i}$, $\hat{j}$, and $\hat{k}$) the gravitational force acting on the tool is

$$\vec{F}_g = \vec{W} = -M_{TS}g\hat{k} \tag{3.55}$$

where $\vec{W}$ is the sensed weight of the tool, g is the acceleration due to gravity ($9.8062\ m/s^2$), and M_{TS} represents the sensed mass of the tool. Here it is important to note the *sensed* mass and *sensed* weight of the tool must be used because the force transducer senses a portion of its own weight when it measures the total external force on the tool (see Section 3.2.7).

Again, from Fig. 3.6 the moment induced by gravity *about the end of the tool* is

$$\vec{M}_g = \vec{r}_{cgs} \times \vec{W} \tag{3.56}$$

In tool coordinates $\vec{r}_{cgs} = -L_{cgs}\hat{a}$, but to perform the cross-product in Eqn. 3.56 $\vec{r}_{cgs}$ must be rotated to world coordinates using

$$[r_{cgs}]_0 = {}^0R_6 \begin{bmatrix} 0 \\ 0 \\ -L_{cgs} \end{bmatrix}_{nsa} \tag{3.57}$$

giving

$$[r_{cgs}]_0 = -a_x L_{cgs}\hat{i} - a_y L_{cgs}\hat{j} - a_z L_{cgs}\hat{k} \tag{3.58}$$

Combining the above with Eqns. 3.55 and 3.56, the gravitational moment becomes

$$[M_g]_0 = a_y L_{cgs} M_{TS} g\hat{i} - a_x L_{cgs} M_{TS} g\hat{j} \tag{3.59}$$

Substituting Eqns. 3.55 and 3.59 into Eqn. 3.54 produces the equation used to calculate interface force

$$[F_{int}]_0 = -{}^0R_6\,[F_{totext}]_{nsa} + M_{TS}g\begin{bmatrix} 0 \\ 0 \\ -1 \\ a_y L_{cgs} \\ -a_x L_{cgs} \\ 0 \end{bmatrix}_0 \tag{3.60}$$

where values of $\vec{F}_{totext}$ are obtained from Eqn. 3.38.

3.2.6 Determining Calibration Constants. The values of the six calibration constants must be determined before Eqn. 3.38 can be used to calculate the total external force. The calibration constants are used to correct for constant force measurement errors included in the measured force *in the sensor coordinate frame*. It is important to understand the calibration values will only be constant when expressed in the sensor frame. If expressed in the world frame they will be a function of the arm joint angles, and if expressed in the tool frame the translation of the moments between the tool and sensor frames must be accounted for. Therefore, it is easiest to calibrate the measurements before they are moved from the sensor frame.

If a known value of $\vec{F}_{int}$ is applied by the arm, and the arm's position is known, Eqn. 3.60 can be used to derive the relationship for calibrating the arm. The easiest way to create a known $\vec{F}_{int}$ is to move the tool to a point in space where it is not in contact with the environment. Then $\vec{F}_{int} = 0$, and the left side of Eqn. 3.60 goes to zero. Next, if the arm is positioned with the sensor and world frames aligned, ${}^0R_6$ becomes an identity matrix, and it is possible to solve for $\vec{F}_{totext}$ without having to perform additional calculations involving the inverse of ${}^0R_6$. So with a properly positioned arm, Eqn. 3.60 is

$$0 = \left[\begin{array}{ccc|ccc} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{array}\right] [F_{totext}]_{nsa} + M_{TS}g \begin{bmatrix} 0 \\ 0 \\ -1 \\ a_y L_{cgs} \\ -a_x L_{cgs} \\ 0 \end{bmatrix}_0 \tag{3.61}$$

But $a_x = a_y = 0$, and if $\vec{F}_{totext}$ is replaced by the results of Eqn. 3.42, then performing the negative identity matrix multiplication reduces the above to

$$0 = \begin{bmatrix} -F_{totext,x} \\ -F_{totext,y} \\ -F_{totext,z} - M_{TS}g \\ -M_{totext,x} - L_{TS}F_{totext,y} \\ -M_{totext,y} + L_{TS}F_{totext,x} \\ -M_{totext,z} \end{bmatrix}_{nsa} \tag{3.62}$$

Substituting for $\vec{F}_{totext}$ and $\vec{M}_{totext}$ with Eqn. 3.38 introduces the calibration constants $\vec{F}_c$ and $\vec{M}_c$ resulting in

$$0 = \begin{bmatrix} F_{c,x} - F_{m,x} \\ F_{c,y} - F_{m,y} \\ F_{c,z} - F_{m,z} - M_{TS}g \\ M_{c,x} - M_{m,x} - L_{TS}(F_{m,y} - F_{c,y}) \\ M_{c,y} - M_{m,y} + L_{TS}(F_{m,x} - F_{c,x}) \\ M_{c,z} - M_{m,z} \end{bmatrix}_{nsa} \tag{3.63}$$

Measured values of $\vec{F}_m$ and $\vec{M}_m$ are available in the sensor frame from the force sensor. These can be used if the above equation is transformed from the tool frame to the sensor frame using the technique of Section 3.2.4 (with the sign on L_{TS} reversed since the translation is in the opposite direction). The transformation is

$$0 = \left[\begin{array}{ccc|c} & I & & 0 \\ \hline 0 & -L_{TS} & 0 & \\ L_{TS} & 0 & 0 & I \\ 0 & 0 & 0 & \end{array}\right] \left[\begin{array}{l} F_{c,x} - F_{m,x} \\ F_{c,y} - F_{m,y} \\ F_{c,z} - F_{m,z} - M_{TS}g \\ M_{c,x} - M_{m,x} - L_{TS}(F_{m,y} - F_{c,y}) \\ M_{c,y} - M_{m,y} + L_{TS}(F_{m,x} - F_{c,x}) \\ M_{c,z} - M_{m,z} \end{array}\right]_{nsa} \tag{3.64}$$

At this point simple algebra leads to the values of the calibration constants

$$\left[\begin{array}{c} F_{c,x} \\ F_{c,y} \\ F_{c,z} \\ M_{c,x} \\ M_{c,y} \\ M_{c,z} \end{array}\right]_{Sensor} = \left[\begin{array}{l} F_{m,x} \\ F_{m,y} \\ F_{m,z} + M_{TS}g \\ M_{m,x} \\ M_{m,y} \\ M_{m,z} \end{array}\right]_{Sensor} \tag{3.65}$$

Again, this assumes the sensor and world frames are aligned, and the tool is not applying any force to the environment. In practice these conditions were satisfied by moving the robot to the ready position, $\vec{q}^T = [0, -90, 90, 0, 0, 0]$ degrees, for each calibration of the force sensor.

3.2.7 Determining Sensed Mass. The force sensor strain gages are mounted on a force sensing element within the force sensor transducer. The weight of any force sensing element mass on the tool side of the strain gages will appear as a force in the force sensor measurements. Based on this, the *sensed* tool mass is defined as

$$M_{TS} = M_{Tool} + M_{SS} \tag{3.66}$$

where M_{Tool} is the tool mass and M_{SS} is the sensed portion of the force transducer mass. This mass must be included as part of the tool mass for purposes of eliminating the gravitational forces and moments from the force measurements.

Values for M_{ss} were not provided by JR3, so a simple experiment was used to determine them. Equating Eqns. 3.52 and 3.38 gives

$$\vec{F}_m - \vec{F}_c = \vec{F}_{ext} + \vec{F}_g \tag{3.67}$$

Breaking $\vec{F}_g$ into two components, one for the gravitational force on the tool ($\vec{F}_{gt}$) and the other for the gravitational force on the sensed portion of the transducer ($\vec{F}_{gs}$) then

$$\vec{F}_g = \vec{F}_{gt} + \vec{F}_{gs} \tag{3.68}$$

and substituting into Eqn. 3.67 gives

$$\vec{F}_m - \vec{F}_c = \vec{F}_{ext} + \vec{F}_{gt} + \vec{F}_{gs} \tag{3.69}$$

If the tool is not in contact with anything, $\vec{F}_{ext} = 0$ and

$$\vec{F}_m = \vec{F}_c + \vec{F}_{gt} + \vec{F}_{gs} \tag{3.70}$$

The experiment consisted of first aligning the force sensor with the earth's gravity vector. Then gravity acts entirely in the positive $\hat{z}_s$ direction, so Eqn. 3.70 can be written as

$$F_{m,z} = F_{c,z} + M_{Tool}g + M_{ss}g \tag{3.71}$$

Now $F_{m,z}$ can be read from the sensor, M_{Tool} can be measured using a scale, and g is known. Two values are unknown so a second equation is required to find $F_{c,z}$ and M_{ss}. The second equation can be obtained by pointing the sensor and tool straight-up, causing gravity to act in the negative $\hat{z}_s$ direction. Now Eqn. 3.70 gives

$$F_{m,z} = F_{c,z} - M_{Tool}g - M_{ss}g \tag{3.72}$$

Equations 3.71 and 3.72 can now be solved simultaneously to find $F_{c,z}$ and M_{ss}.

3.3 Control of Joints 1, 4, 5, and 6.

As explained in Section 3.1.5, joints 1, 4, 5, and 6 are not needed in the impedance control experiment. They are actively commanded to remain fixed at a joint angle of zero. A proportional-derivative (PD) feedback loop is used to control the position of these joints

$$\tau = K_p e + K_v \dot{e} \tag{3.73}$$

where K_p and K_v are the proportional and derivative error gains respectively. The error (e) is

$$e = q_o - q \tag{3.74}$$

so

$$\dot{e} = \dot{q}_o - \dot{q} \tag{3.75}$$

and since the commanded joint positions and velocities (q_o and $\dot{q}_o$) are both zero

$$e = -q \tag{3.76}$$

and

$$\dot{e} = -\dot{q} \tag{3.77}$$

The control law (Eqn. 3.73) becomes

$$\tau = -K_p q - K_v \dot{q} \tag{3.78}$$

Values of K_p and K_v can be determined for each joint by using the single joint model (i.e. no torque coupling between links) from [16, p. 212]. As shown in [40, pp. 5-6 to 5-11]

$$\begin{aligned} K_p &= J_{min}\omega_n^2 \\ K_v &= 2\zeta\omega_n J_{min} \end{aligned} \tag{3.79}$$

where

ω_n = the desired natural frequency of the response.

Table 3.1. J_{min} Values for the PUMA 560 Wrist [37, p. 1053]

Joint	J_{min}
4	0.18
5	0.15
6	0.18

ζ = the desired response damping ratio.

J_{min} = the minimum effective inertia of the link and motor about the joint axis.

For the PUMA 560 values of J_{min} are given by Table 3.1.

It is possible to use the joints not under impedance control as an "electronic" remote center of compliance (RCC, also referred to as an "active" remote center of compliance). Typically RCCs have been mechanical, spring-linkage devices used to provide a measure of passive compliance to the robot end-effector [58, p. 5]. RCCs are used in some compliant motion tasks in lieu of more sophisticated force-feedback controllers. The electronic RCC is different from these passive mechanical devices because the compliance values can be changed by altering the values of K_p (and also K_v to maintain the damping characteristics). A smaller value of K_p will increase the compliance of a particular joint, reducing the force applied by the manipulator. Thus, small errors in alignment can be accommodated even though they are not along a degree of freedom under the influence of the impedance controller.

3.4 Summary.

In this chapter the fundamental concepts of impedance control were discussed and Hogan's full impedance control was developed. The derivation of the simplified version of this control law was presented, as well as the symbolic equations for the component matrices. An original method for obtaining interface force from the force sensor data was described in detail. These theoretical results form the basis for the compliant motion control environment, and the preliminary use of the impedance control law. The implementation of these equations in software and hardware is discussed in the next chapter.

IV. Implementation

A compliant motion control environment was created using both existing equipment in the AFIT Robotics Laboratory, and new equipment procured for this thesis. The environment uses a Digital Equipment Corporation (DEC) MicroVax III and a DEC PDP 11/73 microprocessor to control a Westinghouse/Unimation PUMA 560 robot manipulator. A JR3 Incorporated, Universal Force Sensor was purchased to provide force and moment information. Also, the AFIT Model Shop fabricated a mockup of an aircraft aerial refueling port and nozzle in order to provide hardware for testing the compliant motion controller.

The compliant motion control environment employs a modular, hierarchial control system. The environment was derived from the existing Robotics and Automation Laboratory Hierarchial Control System developed at the Rensselaer Polytechnic Institute, and implemented at AFIT by Dr. M. B. Leahy [39], [34]. A guiding principle in developing the compliant motion environment was to make maximum use of existing software with the minimum necessary modifications.

The theory of Chapter Three was used to develop the impedance control algorithm implemented in the compliant motion control environment. Although the impedance control technique is in theory applicable to any arm configuration, it has never before been applied to a vertically articulated, gear driven industrial robot like the PUMA. As explained in Chapter Three, it was necessary to tailor the impedance control law to the PUMA. Tailoring the control law simplified the computations and accounted for the large gravity and friction forces present in the dynamics of links 2 and 3 (versus the simplified manipulator of [20]).

The PUMA 560 and force sensor hardware are discussed in Sections 4.1 and 4.2. The refueling port and nozzle mockup are described in Section 4.3. The organization and functions of the hierarchial control system elements are discussed in Section 4.4.

4.1 PUMA 560.

The PUMA 560 is a six degree of freedom manipulator with three large links primarily responsible for tool position, and a three degree of freedom, roll-bend-roll wrist providing tool orientation. Because it has six degrees of freedom, the PUMA can reach any position in the workspace and simultaneously achieve any desired tool orientation (except near the workspace boundaries where the arm's motion is limited). For this study, only links 2 and 3 are controlled by the impedance control law, so the PUMA is restricted to two degrees of compliant motion. Therefore the manipulator can satisfy any two position constraints, as long as they are within the plane of motion of links 2 and 3.

All six PUMA joints are revolute and the links are arranged serially. Payloads, grippers, and tools are attached to a tool mounting flange on the sixth link. The arm is vertically articulated so it must contend with large gravitational loads imposed by the link masses, particularly on joints 2 and 3. In general, motors are mounted remote from the joints so as to better balance the links against gravity. Motor torques are transmitted to the joints using high gear ratio transmissions. While the gearing reduces the output torque requirements for the motor, it does add significant stiction, Coulomb, and viscous friction components to the manipulator dynamics.

Optical encoders attached to the motors provide precise measurements of joint position. Every time the digital control processor's power is turned on these must be calibrated with a special software routine which reads potentiometers at each joint to initialize the optical encoder position. A set of mechanical brakes automatically lock the positions of joints 1, 2, and 3 (the large links) when high power to the arm is disconnected. The wrist joints (joints 4, 5, and 6) do not have any mechanism, except internal friction, to prevent them from moving when arm power is off.

The PUMA is pictured in Fig. 4.1. The link coordinate frames and Denavit-Hartenberg parameters are shown in Fig. 4.2.

4.2 *Force Sensor.*

A JR3 Incorporated, Universal Force Sensor system provides force and moment information referenced to a three axis cartesian coordinate system. The two primary components of the system are a wrist mounted force transducer and a separate support electronics package. The system requires a package of power supplies and three cables; one cable provides power to the support electronics; a second cable transfers power to the force transducer and returns analog data signals to the support electronics; while a third cable transfers discrete data over a parallel interface to the digital control processor. Optionally, two more cables can be used to make serial connections to the digital control processor and/or to a separate video display terminal. Figure 4.3 provides a schematic view of the force sensor system.

The force sensor transducer attaches to the tool mounting flange on the PUMA's sixth link. The transducer mass was measured as 0.3040 kg with dimensions as shown in Fig. 4.4. The transducer mass must be included as part of the payload when determining the PUMA's dynamic behavior, although only a portion of the transducer mass (M_{ss}) must be included as part of the gravitational load felt by the sensor (see Section 3.2.7). This transducer "sensed mass" was found to be 0.0771 kg.

The tool (i.e. the mockup refueling nozzle) is bolted to the transducer. Loads placed on the tool are measured as a combination of three forces and three moments about the sensor coordinate frame. The sensor coordinate frame has an origin at the geometric center of the transducer [24, p. 4-24]. The z-axis extends along the transducer centerline in the tool direction, while the x and y axes are arranged in a right-handed cartesian coordinate system as shown in Fig. 4.4. It is important to

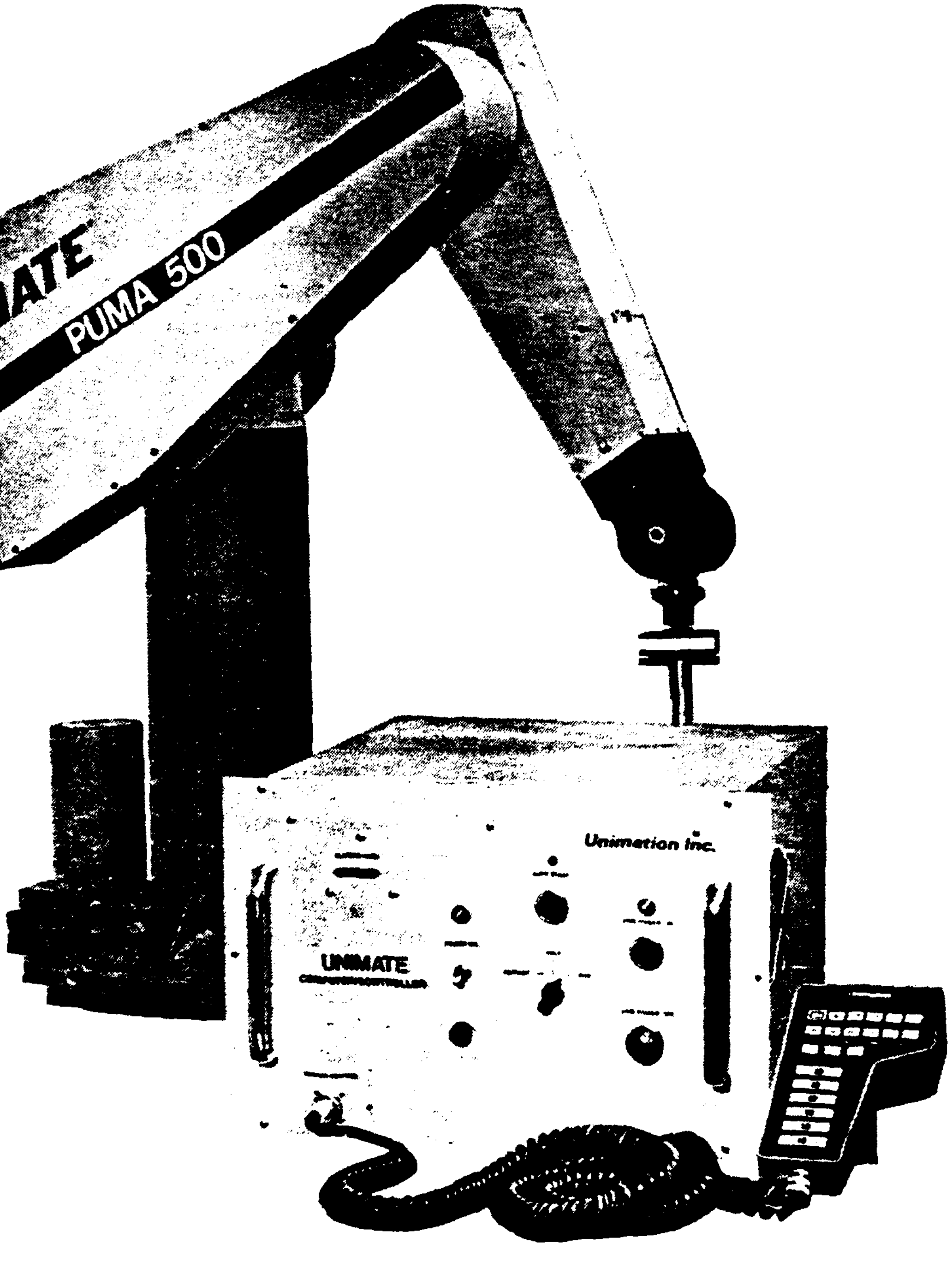

UNIMATE
PUMA 500
Unimation Inc.
UNIMATE

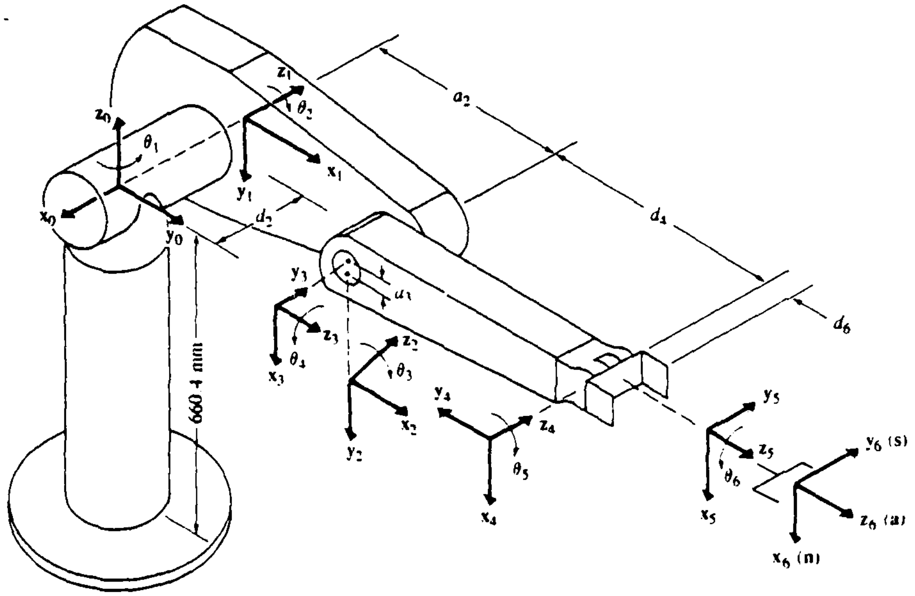
Figure 4.1. PUMA 500 Series Robot Arm with Digital Controller and Teach Pendant [55, p. 1-0].

θ_i is the joint angle from the $\mathbf{x}_{i-1}$ axis to the $\mathbf{x}_i$ axis about the $\mathbf{z}_{i-1}$ axis (using the right-hand rule).

d_i is the distance from the origin of the $(i-1)$th coordinate frame to the intersection of the $\mathbf{z}_{i-1}$ axis with the $\mathbf{x}_i$ axis along the $\mathbf{z}_{i-1}$ axis.

a_i is the offset distance from the intersection of the $\mathbf{z}_{i-1}$ axis with the $\mathbf{x}_i$ axis to the origin of the ith frame along the $\mathbf{x}_i$ axis (or the shortest distance between the $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ axes).

α_i is the offset angle from the $\mathbf{z}_{i-1}$ axis to the $\mathbf{z}_i$ axis about the $\mathbf{x}_i$ axis (using the right-hand rule).



PUMA robot arm link coordinate parameters

Joint i	θ_i	α_i	a_i	d_i	Joint range
1	90	−90	0	0	−160 to +160
2	0	0	431.8 mm	149.09 mm	−225 to 45
3	90	90	−20.32 mm	0	−45 to 225
4	0	−90	0	433.07 mm	−110 to 170
5	0	90	0	0	−100 to 100
6	0	0	0	56.25 mm	−266 to 266

Figure 4.2. Link Coordinates and Denavit-Hartenberg Parameters for the PUMA 560 [16, p. 37]

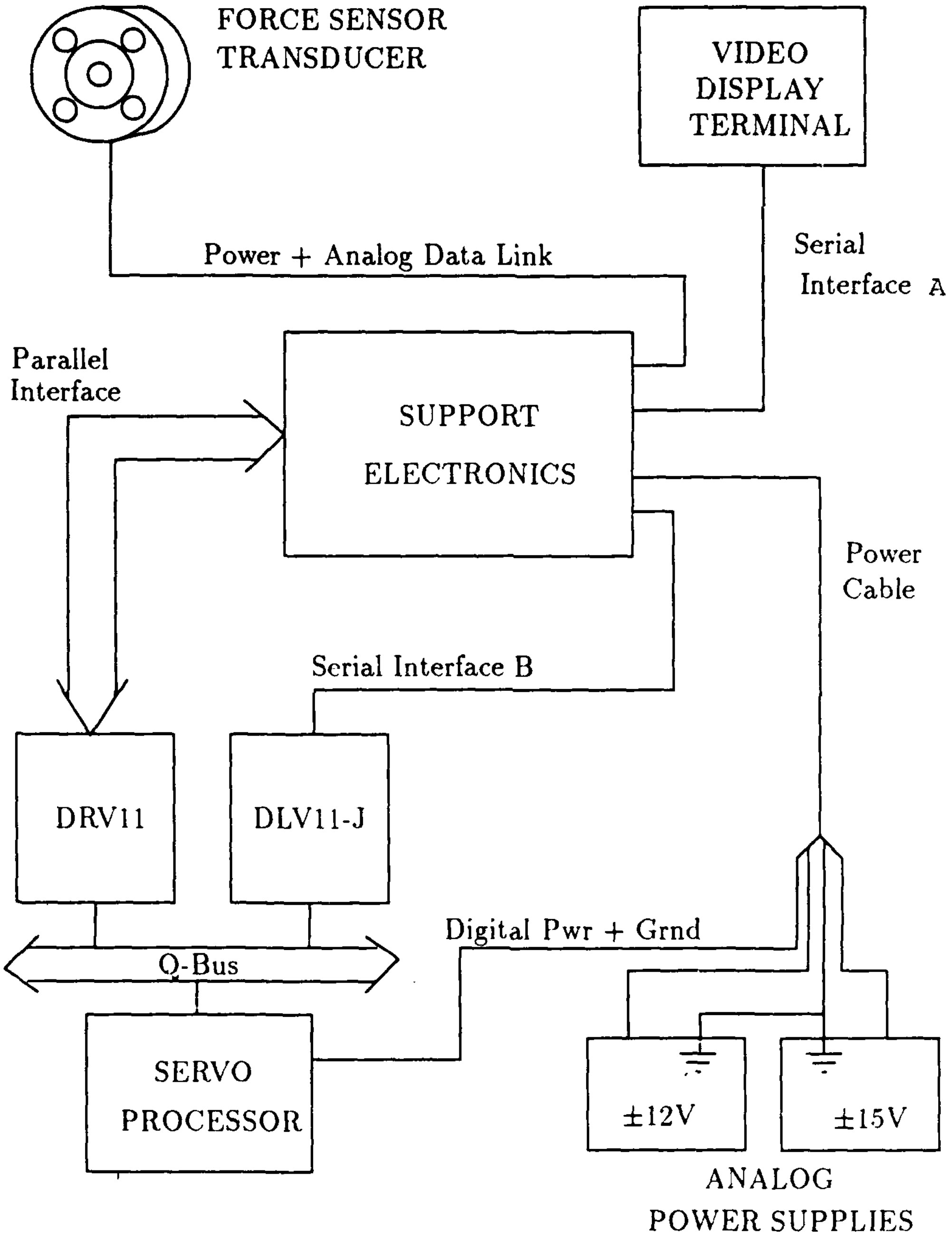

FORCE SENSOR
TRANSDUCER
VIDEO
DISPLAY
TERMINAL
Power + Analog Data Link
Serial
Interface A
Parallel
Interface
SUPPORT
ELECTRONICS
Power
Cable
Serial Interface B
DRV11
DLV11-J
Digital Pwr + Grnd
Q-Bus
SERVO
PROCESSOR
±12V
±15V
ANALOG
POWER SUPPLIES

Figure 4.3. Force Sensor System

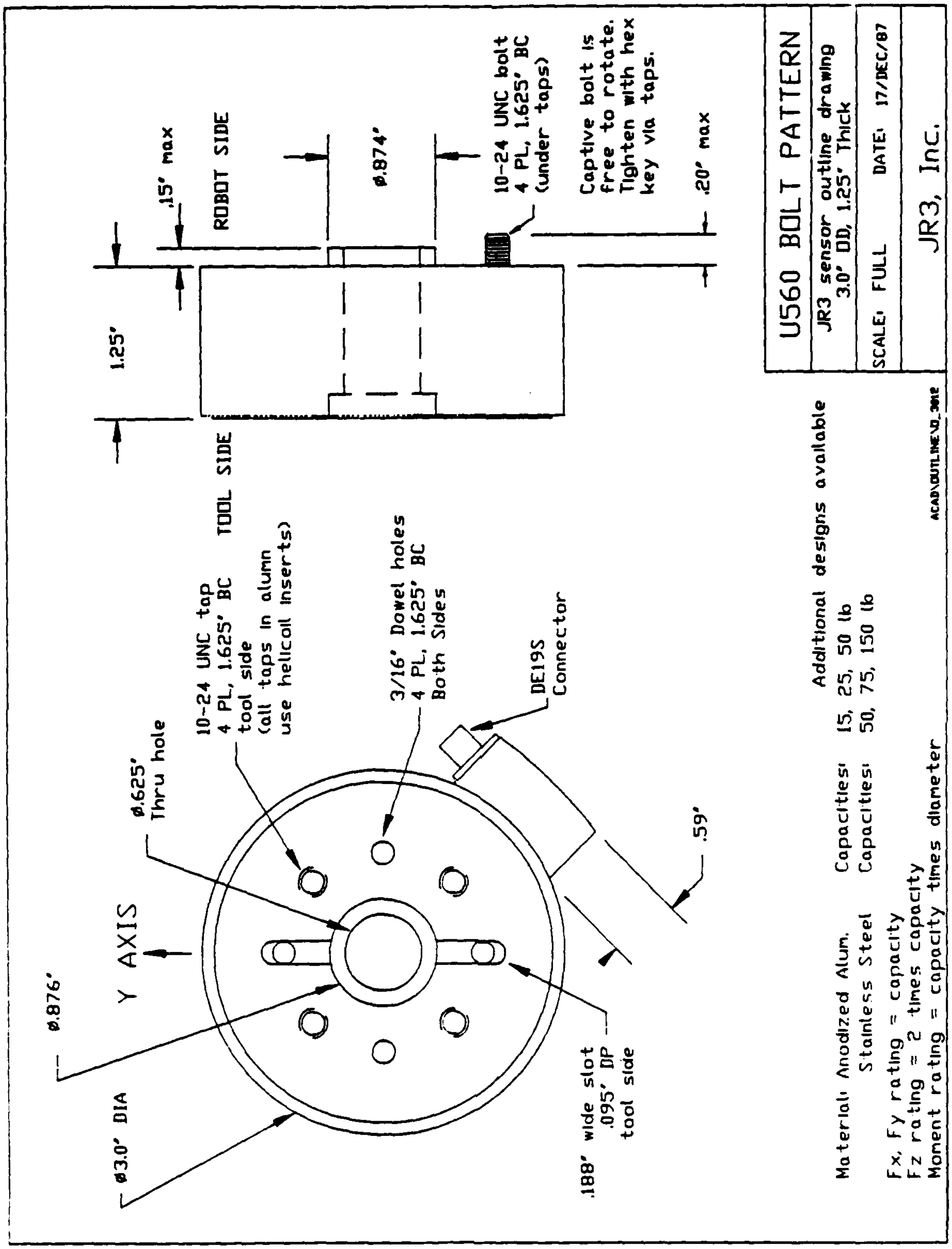

U560 BOLT PATTERN
JR3 sensor outline drawing
3.0" OD, 1.25" Thick
SCALE: FULL
DATE: 17/DEC/87
JR3, Inc.
ACAD\OUTLINE\O_3018
ROBOT SIDE
TOOL SIDE
.15" max
1.25"
ø.874"
10-24 UNC bolt
4 PL, 1.625" BC
(under taps)
Captive bolt is free to rotate. Tighten with hex key via taps.
.20" max
Y AXIS
ø.876"
ø3.0" DIA
ø.625" Thru hole
10-24 UNC tap
4 PL, 1.625" BC
tool side
(all taps in alumn use helicoil inserts)
3/16" Dowel holes
4 PL, 1.625" BC
Both Sides
DE19S Connector
.59"
.188" wide slot
.095" DP
tool side
Material: Anodized Alum.
Stainless Steel
Capacities: 15, 25, 50 lb
Capacities: 50, 75, 150 lb
Additional designs available
Fx, Fy rating = capacity
Fz rating = 2 times capacity
Moment rating = capacity times diameter

Figure 4.4. Force Transducer [24]

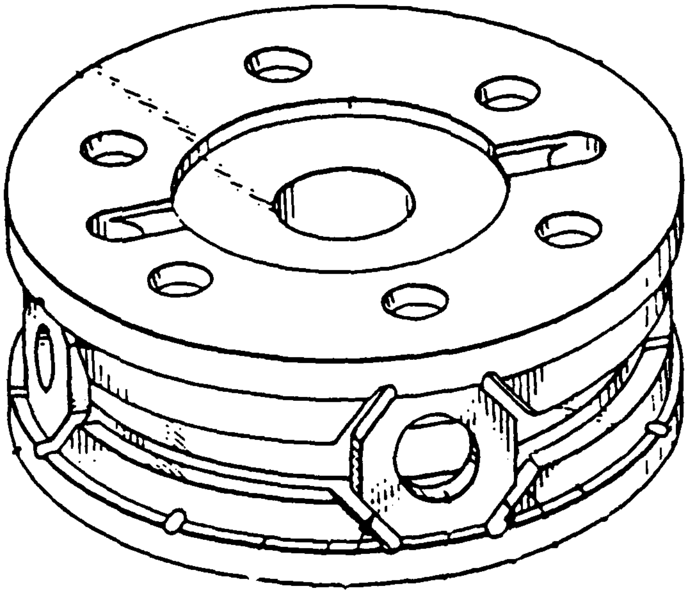


Figure 4.5. Force Sensing Element [48].

note the sensor measures the *external load* (including gravitational forces) placed on the sensor, $\vec{F}_{ext}$, and not the *interface force* applied by the tool, $\vec{F}_{int}$.

The force sensing element of the transducer is housed in the transducer body and is shown in Fig. 4.5. The sensing element consists of two parallel plates connected by four octagonal "bridges". The plates are bolted separately to the PUMA and the tool, so all loads must pass through the bridges. Foil strain gages are mounted on the sides of the octagons, and resistance changes in the strain gages are measured and interpreted as forces or moments using a calibration matrix unique to each transducer (see Section 2.1). In addition to accounting for the relationship between resistance change and force in each gage, the calibration matrix decouples the combined forces and moments sensed by each gage into the three forces and three moments output in the sensor coordinate frame. The relationship between resistance change and applied force is calibrated for a known elastic relationship between stress and strain in the octagonal bridge. Loading the bridges

Table 4.1. JR3 Sensor Force and Moment Limits [24]

Direction	*Limit*
F_x	111.2 N
F_y	111.2 N
F_z	222.4 N
M_x	8.473 N-m
M_y	8.473 N-m
M_z	8.473 N-m

beyond the yield point permanently alters this elastic relationship; requiring recalibration of the gages. Therefore the yield strength of the bridges determines the mechanical strength of the sensor. For this particular JR3 sensor, the mechanical force limits are given in Table 4.1. These limits are used as described in Section 3.2.3 to determine if the transducer is being overloaded.

The support electronics package is the heart of the force sensor system. Signals from the strain gages are amplified and then digitized with a 12 bit A/D converter, while a set of user programmable digital filters suppress noise. The digitized signals are processed and calibrated to reduce the individual strain gage measurements to the six forces and moments in the sensor coordinate frame. Force and moment data are output to the user through either the two RS-423 serial ports, an analog port, a discrete data port, or a direct memory access (DMA) parallel port.

The serial ports can be connected to a digital processor (with a DLV11-J board for the DEC Q-bus) or a stand-alone terminal. The serial ports will send force and moment data, as well as sensor status and configuration information to the user. Commands to change the system configuration, output, digital filters, etc. can be received by the sensor system via the serial ports. Commands for configuring and controlling the system are fully described in [24, pp. 4-1 to 4-25]. In this study a TeleVideo Model 910 video display terminal was connected to serial port A, and was used to verify correct operation of the sensor upon power up.[1]

The analog port provides direct analog output of the force and moment along each coordinate axis. Unfortunately, because the data is calibrated after it is digitized, the force and moment measurements are not decoupled. In a limited experiment this data may be useful to record distinct events, such as impact with an object, but the analog data is of little use here to the digital control algorithm.

The discrete data port has eight command input lines allowing the discrete data port to be programmed by setting a series of single bit "toggles". The discrete data port output consists of eight individual lines. These lines are used to send flags indicating the sensor status, as well as to mark the occurrence of discrete events such as the sensed load exceeding a preset threshold. Discrete data port information is useful in lower level forms of compliant motion control (where force is controlled between certain threshold limits) but was not used in this study since the impedance controller requires more precise force feedback.

Because it allows high speed data transfer, the DMA interface was used to send force data to the digital controller as part of the control loop. Currently, the JR3 software does not permit commands to be received by the sensor system using the DMA interface. The parallel port can only send out force and moment data to the user. Data sent over the parallel link consists of eleven 16 bit words[2]. Words three through eight contain measured values of force and moment in signed integer format [24, p. DMA-1]. The assembly language subroutine GFORCE exercises a DRV11 parallel board to read all eleven words of data. GFORCE uses the standard request handshaking protocol/sequence described in [24, p. DMA-1][3].

---

[1] Correct operation was verified by checking the force/moment offset values and the output of the sensor. If the output seemed reasonable, given the estimated load and the offset values, the sensor was assumed to be operating correctly.

[2] The JR3 Users Manual only indicates 10 words are sent as part of each DMA data package. JR3 admits this is incorrect. Actually, the 10th data word (the "trailer word") is sent twice.

[3] As opposed to the "Optional Protocol for use with DEC DRV-11 Board" also described in the JR3 manual.

About 10ms are required for the sensor system to complete a cycle of converting analog strain gage signals to force and transmitting the force values through the DMA interface. If the digital controller's DRV11 attempts to read information from the sensor before the next sample period, the sensor will ignore the request and GFORCE will generate a timeout error. The actual sample time of the force sensor is a function of the extent of the internal sensor data processing. The sensor system slows down as more complex digital filtering and coordinate transformations are programmed by the user. Future JR3 software releases will allow the user to monitor the sensor processing rate [24, p. 4-23].

The force sensor system requires +5V at 2A to run the digital side of the support electronics, ±12V at 75mA for serial communication, and ±15V at 300mA to run the strain gages. The +5V digital power and digital ground must be tapped off the digital controller (PDP 11/73) to insure a common ground for the DMA interface. The strain gage power is provided by a Hewlett-Packard 6502B Dual DC Variable Voltage Power Supply. The serial communication power is provided by separate ±12V/±5V, single ground, power supply. The analog power supply (±15V and ±12V) grounds are jumpered together to provide a common analog ground, while the separate sides of the HP-6502B are stacked together to provide both +15 and -15 volts with a single ground. The power supply network is detailed in Fig. 4.6.

4.3 Refueling Port and Nozzle Mockup.

Mockups of an aircraft aerial refueling port and the corresponding aerial refueling nozzle were designed and fabricated. They provide a representative surface and tool for testing the compliant motion control techniques. The tool and surface hardware form a physical environment supporting both generalized testing of compliant motion algorithms, as well as a scale demonstration of robotic aircraft refueling.

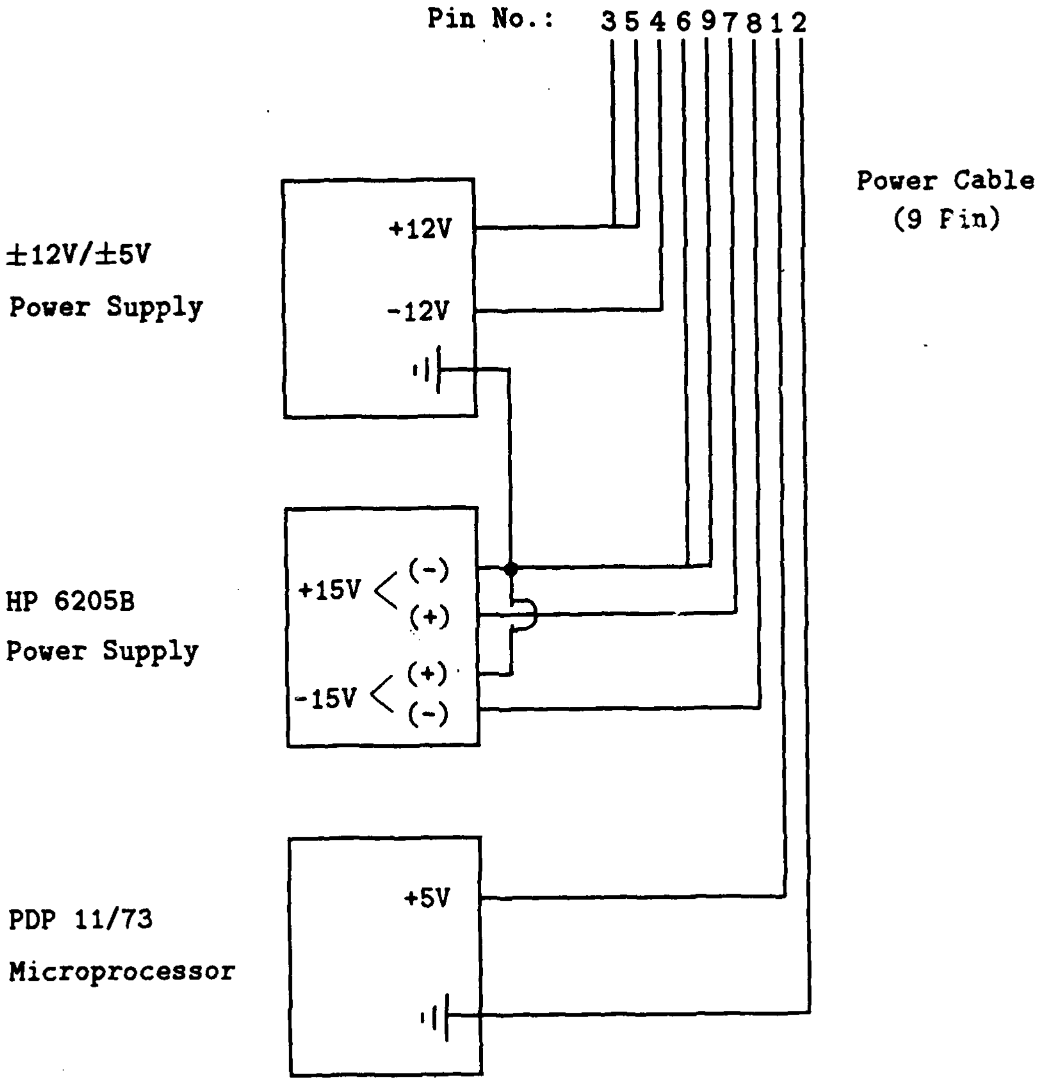

Pin No.: 3 5 4 6 9 7 8 1 2
Power Cable (9 Pin)
±12V/±5V Power Supply
+12V
-12V
HP 6205B Power Supply
+15V
(-)
(+)
-15V
(+)
(-)
PDP 11/73 Microprocessor
+5V

Figure 4.6. Force Sensor Power Supply Network

The refueling port mockup is a half scale "approximate" model of the Universal Aerial Refueling Receptacle Slipway Installation (UARRSI), commonly found on recent USAF aircraft (see Appendix B). The mockup accurately reflects the UARRSI slipway and receiver port dimensions, but it does not have a tapered cylindrical path immediately in front of the receiver port (see Fig. 4.7 and 4.8). Lack of a funnel-like path simplified construction of the mockup, while the squared-off shape actually provides a more challenging test for the compliant motion controller. The UARRSI mockup is mounted in a flat aluminum platform supported by four legs. The aluminum surface of the platform and slipway have been sand-blasted to achieve a low friction surface, just like the actual UARRSI.

The platform has sufficient open area surrounding the slipway to allow general tests of compliant motion controllers on a flat surface. The ability of the controller to negotiate corners and surfaces of varying slope can be tested by moving the tool over the edges of the slipway sides or ramp. Insertion of the tool in the receiver port provides a challenging test of the controller's ability to perform compliant motion tasks requiring precise tool orientation to avoid jamming. The 0.25 inch thick aluminum sheet used to construct the platform presents a very stiff surface to the compliant motion controller. A softer, less demanding environment can be created by replacing the platform holding the refueling port with a two inch thick sheet of high density foam. A two inch thick hardwood laminate platform with rectangular cut-outs also exists. This can be used for testing the controller's ability to perform peg-in-the-hole insertions on a flat, relatively simple, high stiffness surface.

The mockup aerial refueling nozzle simulates the nozzle on the end of a tanker aircraft refueling boom (see Appendix B). The only portion of the mockup nozzle needing to accurately model the actual nozzle is the cylindrical portion which inserts into the receiver port (see Fig. 4.9). The mockup nozzle cylinder is half the size of the actual nozzle, although the diameter has been slightly altered to provide a less demanding initial demonstration of robotic refueling. The altered diameter

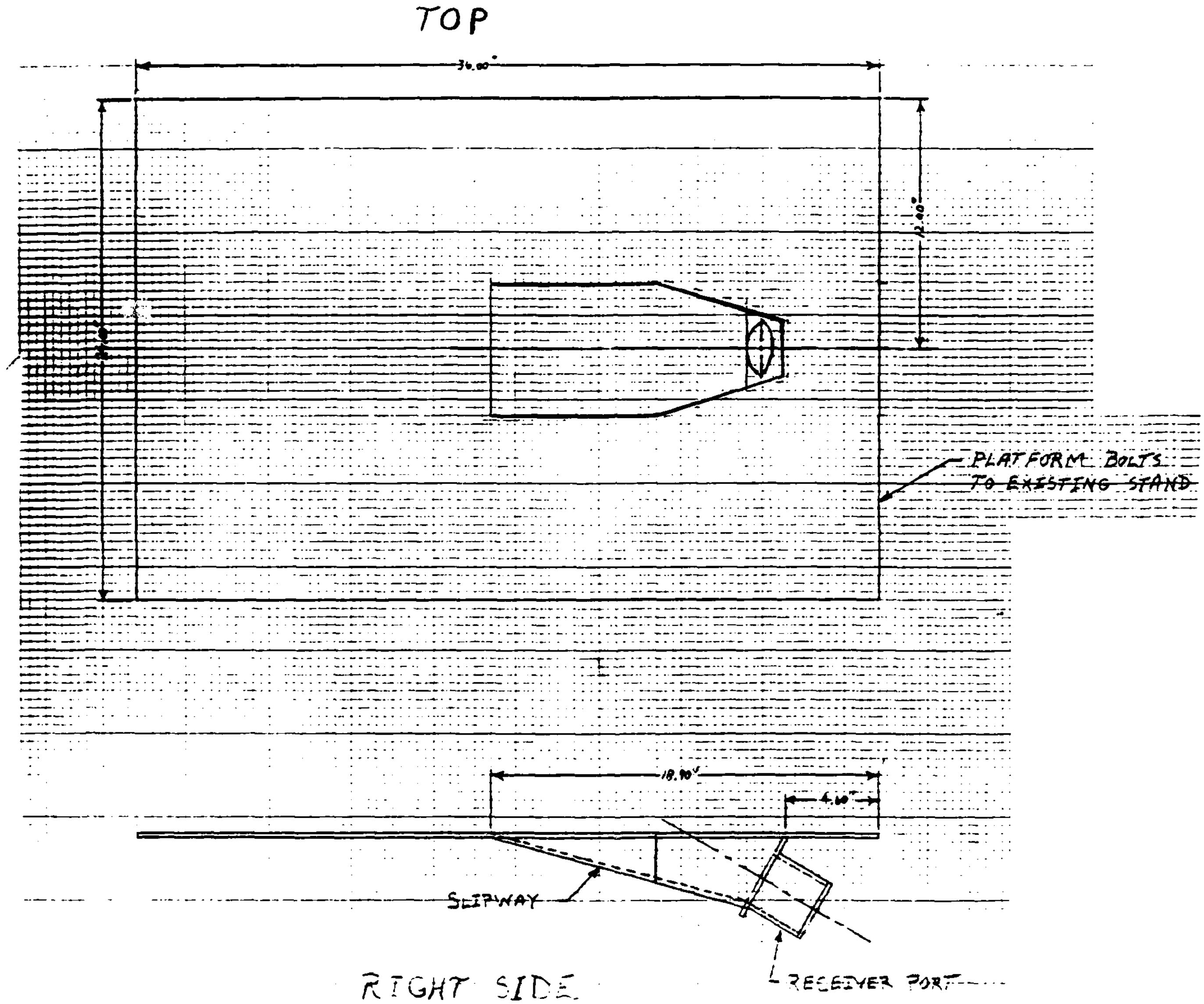

TOP
36.00"
12.00"
PLATFORM BOLTS TO EXISTING STAND
18.90"
4.10"
SLIPWAY
RIGHT SIDE
RECEIVER PORT

Figure 4.7. Refueling Port Mockup and Platform

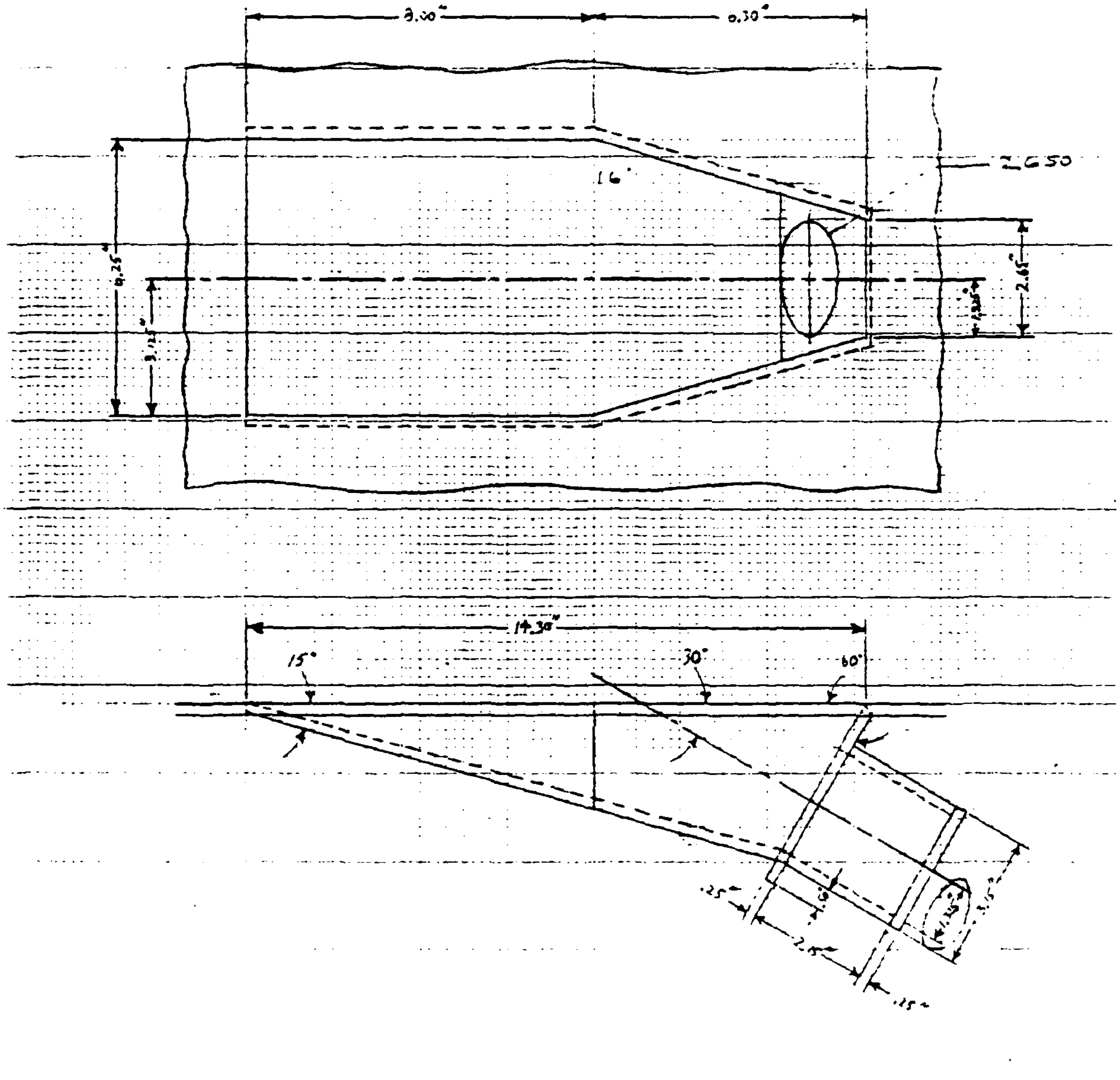

3.00"
6.30"
16°
2.650
6.25"
3.125"
2.65"
1.325"
14.30"
15°
30°
60°
.25
.6
2.5
.15
1.325
3.5

Figure 4.8. Refueling Port Mockup, Detailed View.

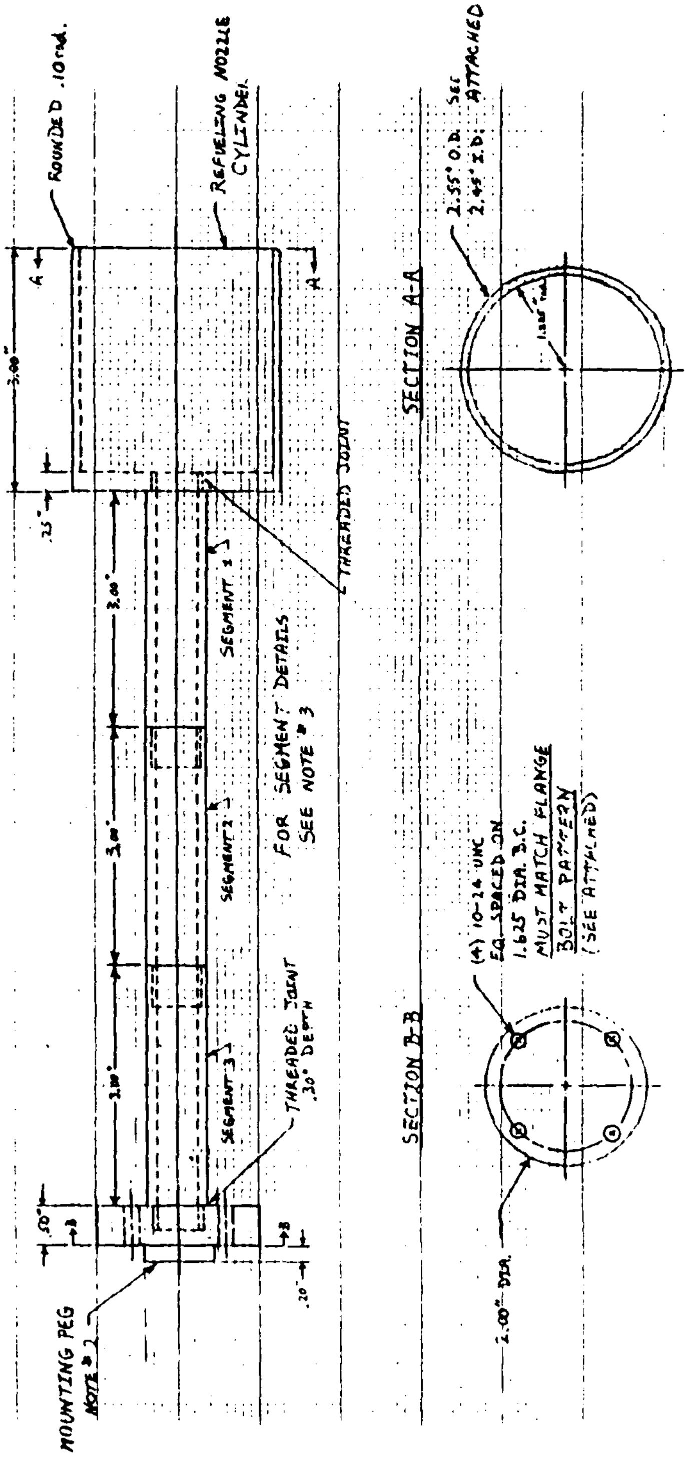

ROUNDED .10 rad.
REFUELING NOZZLE CYLINDER
3.00"
.25"
A
A
THREADED JOINT
SECTION A-A
2.55" O.D. SEE
2.45" I.D. ATTACHED
1.225" rad.
3.00"
SEGMENT 1
3.00"
SEGMENT 2
FOR SEGMENT DETAILS
SEE NOTE # 3
3.00"
SEGMENT 3
THREADED JOINT
.30" DEPTH
SECTION B-B
(4) 10-24 UNC
EQ. SPACED ON
1.625 DIA. B.C.
MUST MATCH FLANGE
BOLT PATTERN
(SEE ATTACHED)
.50"
B
B
.20"
MOUNTING PEG
NOTE # 2
2.00" DIA.

Figure 4.9. Nozzle Mockup

Table 4.2. Tool and Payload Parameters

Parameter	*Value*
$M_{payload}$	0.6034 kg
M_{TS}	0.37651 kg
L_{Tool}	0.34925 m
L_{TS}	0.33338 m
$L_{cg,payload}$	0.09702 m
L_{cgs}	0.20138 m

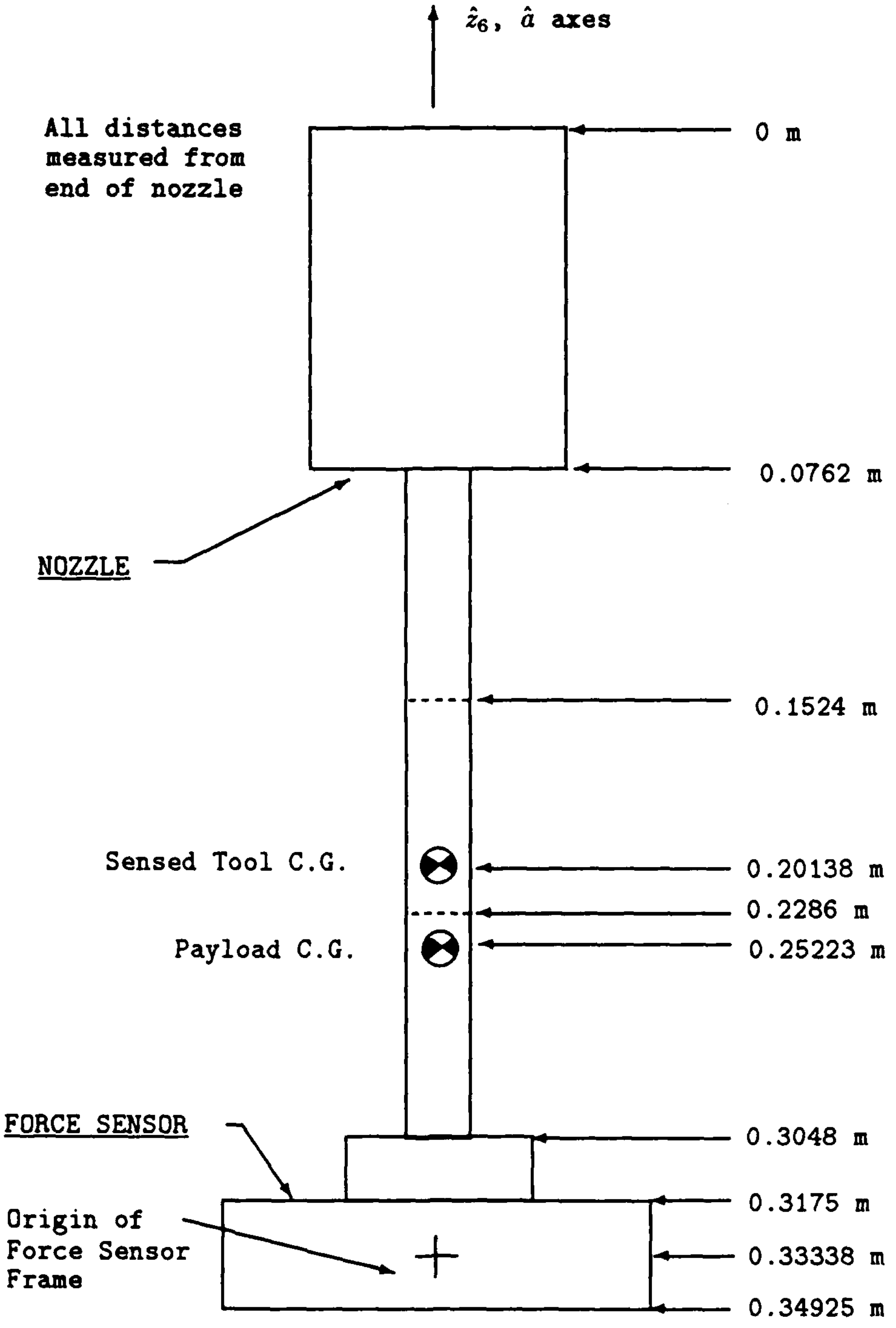
of the mockup nozzle provides a nozzle to receiver port clearance radius of 0.05 inch. If scaled up to full size this would be twice as large as the actual clearance. A more realistic clearance can be obtained by fabricating a new cylinder segment for the tool, or by using tape to increase the outer radius of the nozzle cylinder.

The rest of the nozzle mockup, or tool, consists of a multi-segmented shaft connecting the cylinder to a mounting plate. The mounting plate can be attached to the force sensor or to joint 6 of the PUMA arm. The shaft is threaded and screws into the mounting plate. The overall tool length can be changed by adding or deleting any of the three inch long, threaded shaft segments. Use of a shorter tool decreases the moment applied to the sensor by contact forces at the tool tip, preventing the force transducer from being overloaded. However, the full nine inch shaft is needed to provide sufficient clearance between the PUMA wrist and refueling platform when the nozzle is fully inserted in the receiver port.

The PUMA's entire payload consists of the nozzle mockup bolted to the force sensor transducer. The center of gravity locations and tool length are shown in Fig. 4.10. The tool and payload parameters discussed in Chapter Three and used in the control law are summarized in Table 4.2.

4.4 Hierarchial Control System

The compliant motion control environment uses a hierarchial controller architecture consisting of organizer, coordinator, and hardware levels (see Fig. 4.11).



$\hat{z}_6$, $\hat{a}$ axes
All distances measured from end of nozzle
0 m
0.0762 m
NOZZLE
0.1524 m
Sensed Tool C.G.
0.20138 m
0.2286 m
Payload C.G.
0.25223 m
FORCE SENSOR
0.3048 m
0.3175 m
Origin of Force Sensor Frame
0.33338 m
0.34925 m

Figure 4.10. Centers of Gravity and Tool Lengths. All distances measured from end of nozzle (not to scale).

The organizer provides the user interface and top-level task control. The coordinator executes organizer level instructions to create commands for the hardware level. At the hardware level the control commands from the coordinator are executed by the PUMA, and information about the state of the robot and task are gathered for feedback to the coordinator.

Sections 4.4.1 and 4.4.2 describe the organizer and coordinator levels in detail. Hardware level software to implement commands on the PUMA is considered proprietary by Unimation and is not discussed. Hardware level functions of the force sensor have been described in Section 4.2 and are documented in detail in [24].

4.4.1 *Organizer Level* Organizer level functions are

- User interface
  - Select trajectories
  - Provide positive control of arm movement
  - Signal completion of tasks
  - Display error messages
- Communicate with coordinator level
  - Establish communication link
  - Load coordinator programs
  - Load trajectories
  - Receive and store post-test data
- Execute top-level control algorithm
- Calculate constants for coordinator level routines
  - Constants used for calculating $\vec{F}_{int}$

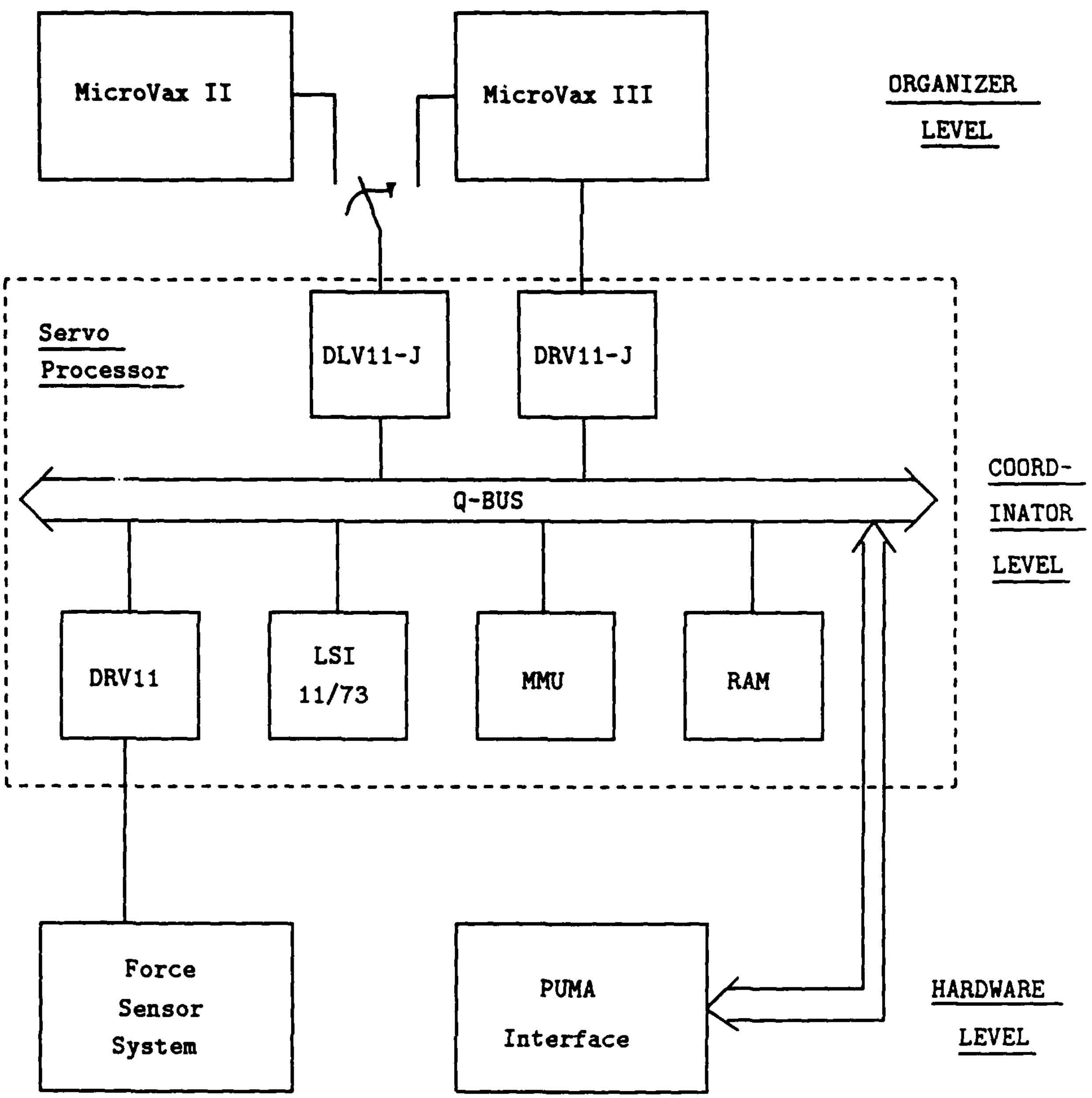

MicroVax II
MicroVax III
ORGANIZER LEVEL
Servo Processor
DLV11-J
DRV11-J
Q-BUS
COORD-INATOR LEVEL
DRV11
LSI 11/73
MMU
RAM
Force Sensor System
PUMA Interface
HARDWARE LEVEL

Figure 4.11. Hierarchial Control System Hardware

- Constants used for calculating torque commands

Either a MicroVax II or a MicroVax III can be switch selected to serve as the organizer level host computer. Both MicroVaxes provide graphics workstation interfaces to the user. Since the MicroVaxes also host Fortran and PDP assembly language compilers, as well as MATRIXX, they form a complete environment for developing, running, and evaluating the performance of compliant motion controllers.

At the organizer level, the main module of the compliant motion environment is the Fortran routine FORCER3AGE. The overall organizer control and communication structure is very similar to the existing "Robotics and Automation Laboratory (RAL) Hierarchial Control System/RAL Real-time Robotic Algorithm Exerciser" (abbreviated RHCS/R3AGE) control environment described in [34], [33]. Existing Fortran subroutines, making extensive use of VMS Q$IO calls, allow the organizer to communicate with the coordinator. Communication occurs over a serial link which can be switched between the MicroVax II and the MicroVax III, depending on the choice of computers for the organizer. An existing Fortran subroutine calibrates the PUMA arm at the beginning of each test series. Derivatives of existing subroutines select test trajectories and receive post-test data from the coordinator. Table 4.3 summarizes subroutines used by the organizer.

Extensive use of existing software and the derivation of FORCER3AGE from the existing RHCS/R3AGE system was desired. A derivative of the RHCS/R3AGE system is the primary system used to control the PUMA at AFIT, mostly for studies of dynamics-based position control. Deriving the compliant motion control environment from the RHCS/R3AGE system presents the user with a common interface using familiar test options and menu choices. Using existing subroutines as much as possible created a modular program structure having a high degree of commonality with the RHCS/R3AGE environment. For a user experienced with the current AFIT PUMA control system, this preserves their familiarity with the

Table 4.3. Organizer Subroutines [39, p. 45]

FUNCTION/NAME	TYPE	COMMENTS
Communications		
DLOAD	E	Download programs to coordinator
PDPINO	E	Interlevel data transfer
PDPCOM	E	Interlevel communication
FORREOUT	D	Receive post-test data from coordinator
ADOUT	E	Receive additional post-test data from coordinator
OFRVAX	E	Handle PUMA arm out of range errors
Calibration		
PUMACAL	E	Calibrate PUMA arm
Trajectory Selection		
SLCTTRJ	D	Select compliant motion trajectory
SLCTIC	D	Select initial test position
Calculate Constants		
CALIBCON	N	Calculate constants used in finding $\vec{F}_{int}$
ADDCON	N	Calculate constants for control algorithm

E = Existing Software, D = Derivative Software, N = New Software

software, and should improve their understanding of the compliant motion control environment.

4.4.2 *Coordinator Level.* The coordinator is the nerve center of the hierarchial control system. The coordinator functions as the digital controller portion of the control system [22, pp. 4–17]. All instructions to control the robot must pass to this level, and be executed to create commanded currents for the PUMA motors, which are then passed to the robot. Information about the task progress, such as joint positions and force measurements, must be gathered from the hardware level and used in the control algorithm. All this must happen in real-time, at a sufficiently high sample rate to guarantee good performance. Because of the heavy computational burden placed on the coordinator, two separate digital processors are used. Labelled the Servo and Parallel processors, they allow asynchronous execution of different portions of the control loop. The Servo processor acts as the *communication* and task control node at the coordinator level. The Parallel processor acts to lighten the Servo processor's load by performing a portion of the control algorithm computations.

4.4.2.1 *Servo Processor.* The Servo processor is a PDP 11/73 microprocessor with a memory management unit (MMU), and random access memory (RAM) available on the Q-bus. The Servo processor communicates with the PUMA arm's digital interface over the Q-bus. The Servo processor also communicates with the force sensor system using a DRV11 card linked to the force sensor's direct memory access port. Communication with the organizer level is performed using a DLV11-J serial card. If the MicroVax III acts as the Parallel processor, then communication is by means of a DRV11-J card.

Coordinator functions are

- Communicate with the organizer

  - Receive programs
  - Receive trajectories
  - Send post-test data
  - Receive commands to begin task execution
  - Send task completion messages
- Communicate with the Parallel processor
  - Send joint angles and end of trajectory flag
  - Receive impedance coefficients
- Communicate with the PUMA
  - Send out current commands
  - Read joint angular position
  - Check for communication errors
  - Stop arm motion
- Obtain interface force
  - Read force data from force sensor
  - Check for communication errors
  - Convert force data to interface force
  - Check for excess force
- Evaluate the control law to determine motor current commands
  - Interpolate the trajectory
  - Calculate the PD control law torques for joints 1, 4, 5, and 6
  - Calculate impedance control law torques for joints 2 and 3

    - Convert torque commands to current
- Supervise algorithm timing
    - Obtain interrupt timing signal from the PUMA
    - Control sample rates for
        - Parallel processor
        - Force sensor
        - Updating motor current commands
- Receive and execute programs to calibrate the PUMA and force sensor

Software for execution on the Servo processor is written in PDP assembly language and compiled with a RSX11 compiler. Where possible, existing coordinator level software is used. In some cases existing routines are used with minor modifications. The assembly language program IMPSER2 is the main routine for implementing the impedance control algorithm on the Servo processor. Table 4.4 lists subroutines used by the Servo processor.

Using assembly language maximizes the software execution speed. Equations were symbolically reduced to further improve the processing speed. For the equations used in calculating $\vec{F}_{int}$ and the impedance control torques (Eqns. 3.60 and 3.21), this requires the matrix equations to be broken into the individual algebraic equations with elimination of all the "zero multiplies." Where possible, the terms of the equations are arranged to bring all the coefficients of a particular variable together as a single constant, which is then calculated at the organizer level and sent to the coordinator prior to the test run.

Existing RHCS/R3AGE software was used to communicate with the organizer level MicroVax and the parallel processor. Only the subroutine for sending post-test data to the organizer required modification. Similarly, existing software is used, without modification, to communicate with the PUMA. However, it was

Table 4.4. Coordinator Subroutines

FUNCTION/NAME	TYPE[4]	COMMENTS
Communications		
INITDR	E	Initialize DRV11-J
RDVAX	E	Read data from organizer
SDVAX	E	Send data to organizer
PDPVAX	E	Send data to organizer, wait for reply
FHDVAX	D	Send $\vec{F}_{int}$ and $\vec{q}$ post-test data to organizer
SDRV11	E	Send data to Parallel processor
GDRV11	E	Read data from Parallel processor
GFORCE	N	Read force sensor data
COHEAD	E	Prepare Servo processor to receive and execute new programming from the organizer
RREPOS	E	Read joint position from PUMA (radians)
REPOS	E	Read joint position from PUMA (degrees)
WCMODE	E	Send current values to PUMA
Arm Control		
TESTST	E	Move arm to initial test position using VAL mode
ASTOP	E	Stop PUMA motion
HPBOFF	E	Turn arm power off
Force Calculation		
CFORCE	N	Convert data counts to floating point and calibrate
MAXFOR	N	Check for excess force
CFINT	N	Calculate $\vec{F}_{int}$ from measured force
Calculate Control Law		
KIN62	N	Calculate forward kinematics (${}^0R_6$ for 6 degrees of freedom, P_x and P_z for two degrees of freedom)
GEJTRJ	E	Get position and velocity trajectory points
FRICTC	E	Compensate motor current values for friction
Timing		
ETIMER	E	Turn PUMA interrupt clock on
DTIMER	E	Turn PUMA interrupt clock off
Test Preparation		
PFMAGE	D	Calibrate force sensor, move arm to initial position, prepare data buffers for test

E = Existing Software, D = Derivative Software, N = New Software necessary to develop a new subroutine for reading force sensor data over the DMA interface.

The subroutines for calculating interface force ($\vec{F}_{int}$) based on the force sensor data are all new. Although the impedance controller has only been established for two degrees of freedom, the force sensor routines are set up to calculate all six elements of $\vec{F}_{int}$ for the general case of six degree of freedom motion. Then only $F_{int,x}$ and $F_{int,z}$ are used in the impedance control algorithm. Calculating $\vec{F}_{int}$ for the most general case causes little loss in computation speed since the coordinate transformations involved require the use of the full force vector. Therefore, it is almost as quick to get all six elements as it would be to get just the two elements. Setting up the $\vec{F}_{int}$ calculations for the most general case also allows the use of this software in a future n degree of freedom compliant motion controller.

The impedance control algorithm is structured to allow the Parallel processor to calculate all the joint angle dependant quantities used in the control law. The positions of joints 2 and 3 are passed to the Parallel processor while values for the matrix impedance coefficients $IJ^{-1}M^{-1}$, $IJ^{-1}M^{-1}BJ$, $(J^T - IJ^{-1}M^{-1})$, and the vector $\vec{S}(\vec{q})$ are returned (two floating point values sent and fourteen received). For reasonably short sample times and low velocities, the joint positions are assumed to display little change from one sample period to the next. Therefore the values of the impedance coefficients should change very little from one sample period to the next (as long as J is not approaching a singular value), and they need not be updated on every pass through the control algorithm. This permits the Parallel and Servo processors to be run asynchronously. Asynchronous sampling allows a fast, inner control loop to be established on the Servo processor while a slower, outer control loop is used to update coefficient values.

The PUMA's interrupt signal generator is the only source of timing information in the hierarchial control system. Time signals are sent out by the PUMA interface every 7ms, so all Servo control loop sample times must be multiples of

7ms (i.e. 7ms, 14ms, 21ms, 28ms,...). The Servo control algorithm permits the Parallel processor to be accessed only at sample times which are even multiples of the Servo control loop sample time (i.e. if the Servo control loop sample time is 14ms, the Parallel processor may run at sample times of 14ms, 28ms, 42ms,...). Unfortunately, the force sensor requires 10ms to complete its internal computations, so the minimum sampling time for either the Servo or Parallel processors was 14ms.

The Servo processor requires the desired cartesian position trajectory in order to determine the position error term, $\vec{x} - \vec{L}(\vec{q})$, of the impedance control law. Position trajectories are stored in the Servo processor as $n \times 6$ arrays, where n is the number of points in the trajectory. To enable expansion to a full six degree of freedom control law, the arrays are six columns wide to store a full six degree of freedom trajectory. Memory space limits the number of trajectory points to $n \approx 1000$. With a 14ms sample time, trajectories are limited to a 14 second duration if every point in the trajectory is used in immediate succession. Short duration trajectories are not useful for a slow moving robot because long path length assembly tasks may take longer than 14 seconds to complete. Treating the trajectory points as set points and interpolating additional points between them allows longer duration trajectories. Therefore a simple, linear interpolation routine is included in the control loop to give the robot the capability for extended duration trajectories. The user can select the number of points to be interpolated between set points.

The interpolation scheme adds another loop to the control algorithm since the controller must periodically go to the RAM and acquire the next position set point. As part of this loop, the algorithm saves $\vec{F}_{int}$, $\vec{q}$, and the x and z cartesian positions (P_x and P_z). These three sets of values are sequentially appended to values saved on previous passes to create force, joint position, and cartesian position history buffers covering the entire test duration. The history buffers can be sent to the organizer as the post-test data package. Saving data values only as part of the interpolation loop limits the amount of data stored in RAM to an acceptably small level.

IMPSER2, the assembly code for the Servo processor's portion of the control algorithm, allows for the expansion of the impedance control law to more than two degrees of freedom. As previously noted, $\vec{F}_{int}$ and the cartesian trajectory information are already set up to handle the general case of six degrees of freedom. To expand IMPSER2 to six degrees of freedom would principally require changes in loop counter values and some pointer statements used in calculating current commands. Larger blocks of memory would also have to be reserved for the larger vectors and arrays present in a six degree of freedom system. The increased number of calculations required for a six degree of freedom impedance control law would be partially offset by eliminating the calculations required for the PD controller on joints 1, 4, 5 and 6.

The impedance control law could also be very easily expanded to allow a commanded cartesian velocity to be included in the control law (the $\vec{v}_o$ term of Eqn 3.20). The organizer passes a $n \times 6$ dummy velocity trajectory to the Servo processor. The Servo processor stores the information in RAM but does not use it in this preliminary, simplified version of the impedance control law. If the impedance controller includes the commanded velocity, the commanded velocity must still be small so as to prevent any significant Coriolis or centrifugal effects. Large commanded velocities would require the $\vec{C}(\vec{q}, \dot{\vec{q}})$ and $\vec{G}(\vec{q}, \dot{\vec{q}})$ functions to be included in the control algorithm, and this would significantly increase the number of computations required.

4.4.2.2 *Parallel Processor* To reduce the computational burden on the Servo processor and speed up coordinator level program execution, a major portion of the impedance control law calculations are performed on separate computer — dubbed the Parallel processor. Both a PDP 11/73 and a MicroVax III have been tested in the Parallel processor role. The MicroVax is preferred since it permits programs to be written in higher level languages such as Fortran, instead of assembly language. The MicroVax also provides its own user interface and disk storage so it is not necessary to establish a communication link with the organizer. The MicroVax III has sufficient computational speed (three million instructions per second) to perform the impedance coefficient calculations within the 14ms sample time. The VAXlab software utility package enables high-speed communication with the Servo processor, allowing the joint angles and impedance coefficients to be passed without significantly slowing the control algorithm execution speed [12], [13].

If the serial communication switch is set to also make the MicroVax III the organizer (see Fig. 4.4), and the user is working at a DEC Graphics Workstation Terminal, then the need for two separate user interfaces is not a problem. The user can open one window on the terminal to run organizer processes while another window is used to simultaneously run the Parallel processor. Since the organizer is quiescent during the time critical portion of the Parallel processor calculations, there is minimal interference between the two processes.

The Parallel processor's functions are

- Communicate with the Servo processor
  - Initialize communication link
  - Receive joint angles and end of trajectory flag
  - Send impedance coefficient values
- Compute constants used for impedance coefficient calculations
  - Read data files from disk
  - Compute constants

- Compute impedance coefficients

The Parallel processor communicates with the Servo processor using VAXlab communication functions to run the DRV11-J card. These functions are very efficient and eliminate the need for any new communications subroutines.

The impedance control law requires the 2×2 matrix products $IJ^{-1}M^{-1}$ and $IJ^{-1}M^{-1}BJ$, as well as the 2×2 matrix sum $(J^T + IJ^{-1}M^{-1})$ and the 2×1 vector $\vec{S}(\vec{q})$. These quantities are termed the impedance coefficients. They can be calculated if the positions of joints 2 and 3 are known. The Parallel processor aids the Servo processor by performing this portion of the control algorithm calculations. The impedance coefficient calculations are ideal tasks for the Parallel processor since they represent a major portion of the total calculations required, and they require only a single 2×1 vector, $\vec{q}$, to be passed from the Servo to Parallel processor.

The elements for the Jacobian and inverse Jacobian (J and J^{-1}) are calculated from Eqns 3.30 and 3.31 respectively. The inertia tensor (I) values are determined using Eqns 3.32, 3.33 and 3.34, and the payload parameters from Table 4.2. Values for the diagonal desired mass (M) and damping (B) matrices are read from a disk data file. The gravitational torques ($\vec{S}(\vec{q})$) are calculated using Eqn 3.35.

The equations for all these components are symbolically simplified by rearranging terms to group multiple individual constants together into fewer, comprehensive constants. These comprehensive constants are evaluated once prior to the start of the test. The impedance coefficients are calculated during the test by evaluating each individual component matrix (I, J, J^{-1}, *etc.*) and then multiplying or adding the matrices together to produce the coefficients. Since the component matrices are fully populated, this approach minimizes the number of calculations required.

Table 4.5. Parallel Processor Subroutines

NAME	**TYPE**[5]	**COMMENTS**
IMPCONST	N	Read parameter values from data file and calculate constants
CIMPG2VAX	N	Calculate impedance coefficients

E = Existing Software, D = Derivative Software, N = New Software

The Fortran program PCIMPVAX is the routine containing the top-level Parallel processor algorithm. The evaluation of the component matrices and their combination into the impedance coefficients occurs in a single subroutine. This subroutine forms the core of the Parallel processor algorithm. PCIMPVAX can be expanded to allow for six degrees of freedom by changing the number of values passed to and from the Servo processor, and by replacing the impedance coefficient calculation subroutine with one designed for six degrees of freedom. The six degree of freedom impedance coefficient calculations would also require the elements of I, J, J^{-1}, M, B, and $\vec{S}(\vec{q})$ to be symbolically determined for the six degree of freedom case. This is not an overwhelming task if MACSYMA is properly used to help with the symbolic manipulation.

Subroutines used by the Parallel processor are listed in Table 4.5.

4.5 *Summary*.

This chapter has discussed the creation of a compliant motion control environment implementing a prototype version of an impedance control law. The robot, force sensing, and test environment hardware have been described. The computer hardware and software forming the hierarchial control system are also described. Detailed listings of the computer software developed for the compliant motion environment are contained in AFIT Robotics Systems Laboratory Report No. 3 [14].

The use of this compliant motion environment to investigate the implementation of the impedance controller is discussed in the next chapter.

V. Evaluation

The compliant motion control environment described in Chapter Four was used to implement and evaluate a preliminary version of an impedance controller. The impedance controller implemented here is only a prototype version used for testing the compliant motion environment, and initial investigations of compliant motion control on the PUMA arm. The initial trajectory tracking tests described in Section 5.3, indicated the impedance controller's capability to control the PUMA in free-space. The compliant motion tests explained in Section 5.4, demonstrated the controller's ability to use active compliance to reduce the interface force when the tool was in contact with the environment. The initial trajectory tests also identified several areas requiring improvements to achieve better controller performance.

Before the trajectory tracking and compliant motion tests could be conducted, it was necessary to evaluate the performance of the force sensor and the software for calculating interface force. Force sensor performance is quantified in Section 5.1. The digital controller's sample time was known to have important effects on controller performance. Therefore, times to execute major components of the control algorithm were measured, and the results are discussed in Section 5.2.

5.1 Force Sensing Performance.

One of the major activities in this thesis effort was incorporating the force sensor into the PUMA's control environment. This effort consisted of developing a model describing the behavior of the JR3 force sensor, integrating the force sensor with the PUMA arm and control system, and creating software to calculate the interface force applied by the arm based on force and moment data from the sensor. Once these tasks were accomplished, it was necessary to test the sensor and software for calculating interface force ($\vec{F}_{int}$) in order to determine the accuracy of the sensing system.

Table 5.1. Sensor Scale Values and Corresponding Resolution

Sensor Axis	F_{scale}	*Resolution*
F_x	111.2 N	0.0271 N
F_y	111.2 N	0.0271 N
F_z	222.4 N	0.0543 N
M_x	8.474 N-m	0.0021 N-m
M_y	8.474 N-m	0.0021 N-m
M_z	8.474 N-m	0.0021 N-m

5.1.1 Force Sensor Resolution.

The resolution of the force sensor, in the sensor frame, is a function of the number of bits used in the A/D converter. The JR3 force sensor uses a 12 bit A/D converter and sends these 12 bits to the PDP 11/73 processor. The resolution is therefore

$$Resolution = \frac{1}{2^{12}} F_{scale} \tag{5.1}$$

In this study F_{scale} was set equal to the rated load capacity of each axis of the JR3 sensor. Given these scaling values, the resolution of the sensor is shown in Table 5.1.

5.1.2 Force Sensing Accuracy.

The force sensor and associated software for determining $\vec{F}_{int}$ in world coordinates were tested by positioning the PUMA arm so the tool was not in contact with any object. Since the only force acting on the tool was gravity, and the gravity force and moment were compensated for in the calculation of $\vec{F}_{int}$, the resulting values for $\vec{F}_{int}$ should be zero. The variation of the actual output values from zero served as a measure of the accuracy of the force sensor's measurements.

Results of these force sensor tests are shown in Table 5.2 for four different arm positions (Cases 1 through 4), and two different sets of calibration values (Run 1 and Run 2). The force sensor had the least amount of error for Case 1, because the arm was returned to the position used to calibrate the force sensor (the ready position). Cases 2 and 4 both showed comparable error values, none of which were

Table 5.2. F_{int} Error Data from the Force Sensor Accuracy Tests

Case	*Joint Position (degrees*	*Run*	*Error (N and N-m)*					
			F_x	F_y	F_z	M_x	M_y	M_z
1	0,-90,90,0,0,0	1	0.054	-0.054	0	-0.018	-0.018	-0.01
		2	0	0	0	0	0	0
2	0,-90,0,0,0,0	1	-0.326	0.326	0.760	-0.029	-1.200	0.113
		2	-0.222	0.979	0.271	-0.022	-1.030	0.359
3	40,-120,-15,-25,63,75	1	-0.170	3.817	-4.760	0.767	1.150	1.545
		2	-0.688	3.505	-5.752	0.814	1.539	1.364
4	40,-120,-15,0,63,0	1	0.215	0.174	0.689	-0.720	-0.865	0.097
		2	0.620	0.120	0.122	-0.579	-0.753	0.180

unreasonably high. Case 3 had the worst errors because it involved rotations of joint 4 and 6 away from the calibration position (zero degrees). These rotations caused problems due to poor calibration of the PUMA's position encoders for joints 4 and 6. Since it had the worst errors, and because it involved the most general rotation of the arm, Case 3 was used to indicate the accuracy limits in calculating $\vec{F}_{int}$. For purposes of setting deadband limits in the control algorithm, $\vec{F}_{int}$ was assumed to be accurate to within ± 6.0 N of axial force and ± 2.0 N-m of moment.

5.1.3 Force Sensing Error Sources. The principal source of error in determining $\vec{F}_{int}$ was the calibration accuracy of the PUMA's joint encoders. Errors in encoder calibration prevented the arm from properly orienting itself within the world coordinate frame. As a result, the force sensor and world frames were not accurately aligned when the force sensor was calibrated, so the force sensor calibration values were in error. Later, when the arm positioned the tool at some arbitrary orientation in the workspace, there were errors in the ${}^0R_6$ coordinate transformation matrix. These errors in ${}^0R_6$ prevented an accurate transformation from tool to world coordinates of the measured force values, and as a result the compensation for gravitational force on the tool was incorrect.

Other minor error sources are the accuracy in determining the sensed mass of the force sensor (M_{SS}), sensor noise and numerical accuracy, and the computational accuracy within the PDP 11/73. Of these, errors in determining M_{SS} are the most readily fixed. Instead of finding M_{SS} by the experiment described in Section 3.2.7, the force sensor transducer could be disassembled and M_{SS} determined directly using a scale. This technique was not used in this study because of the risk of damage to the force sensor.

5.2 Timing.

The compliant motion control environment is a digital control system using multi-rate sampling [22, p. 89]. Multi-rate sampling allows the control algorithm to be divided into high-speed inner control loops and low speed outer control loops. Different microprocessors can be used for each loop to distribute the computational load among multiple computers, thus decreasing the overall calculation time. Sample time (i.e. the time interval between updates information to/from the robot) and calculation time are directly related. For sampled-data digital control systems a short sample time is desired to improve system performance (see Section 5.2.2).

Initial tests with the impedance control law used a sample time (T_s) of 14 ms (0.014 s). For the Servo processor a 14 ms sample time was necessary to accommodate the force sensor (see Sections 4.2 and 4.4.2.1). The Parallel processor calculations were easily completed in under 14 ms, so this portion of the control algorithm was also sampled with T_s = 14 ms. Performance of the robot with the 14 ms sample time was not entirely satisfactory, and a shorter sample time was considered as one method of improving performance. With the force sensor running at 14 ms, an attempt was made to run the remainder of the Servo processor and Parallel processor routines with a 7 ms sample time. Unfortunately this was not successful, apparently because the Servo processor algorithm could not be completed in less than 7 ms.

The inability to run the controller with a 7 ms sample time prompted an

Table 5.3. Control Algorithm Execution Times

Function	*Execution Time* (ms)
Basic Control Law	7.50
Interpolation	0.70
Force Calculations	1.40
Overhead	0.25
Total	9.85

experiment to estimate the actual times required to complete each portion of the control algorithm. The experiment and results are discussed in Section 5.2.1. The importance of sampling time and its effect on the compliant motion controller design are discussed in Section 5.2.2.

5.2.1 *Timing Tests.* The time required to run the Servo processor impedance control routines was determined by using the MicroVax III in the Parallel processor role and obtaining timing information from the MicroVax system clock. The average time per control loop cycle was found by measuring the time to run the whole trajectory and then dividing by the number of cycles required for the trajectory. Selectively eliminating portions of the algorithm allowed the time to complete the missing portions to be computed from the change in cycle time. The tests were repeated three to four times to produce the average times shown in Table 5.3. The variance of the test data suggests the results are accurate to ±5 percent.

The time to compute the basic control law included the time to:

- Send torque commands and receive joint angle data from the PUMA.
- Send the two joint positions and receive the 14 impedance coefficients from the Parallel processor.
- Compute the PD control law for joints 1, 4, 5, and 6.
- Calculate the next desired position using slope values determined by the interpolation scheme.

- Compute the forward kinematics (including the ${}^0R_6$ matrix required for the force sensor calculations).
- Determine the impedance control torques using the impedance control law (Eqn. 3.21) and the impedance coefficients from the Parallel processor.

A time of 7.5 ms to compute the basic control law is reasonable considering the extent of the operations required. A similar algorithm has been used in the ARCADE environment to implement a six degree of freedom PD controller with a calculation time of less than 7 ms [53]. Although the impedance control law only uses two degrees of freedom, it required more computations on the Servo processor than the PD control law because of the force feedback terms. The impedance controller also performed additional computations to interpolate the desired position, determine the forward kinematics, and implement the PD control law on joints 1, 4, 5, and 6[1]. With all these additional computations, the basic control law can reasonably be expected to take slightly longer than the simpler PD controller.

The 0.70 ms required for interpolation included the time to read the next trajectory set point from memory, and the time to calculate the slope of the line between the current position and the next set point. The 1.4 ms needed to perform the force calculations included:

- The conversion of force sensor output data (A/D counts) to calibrated forces and moments in the sensor frame (CFORCE subroutine).
- Checking these forces to insure they do not exceed the sensor's mechanical limits (MAXFOR subroutine).
- Transforming the forces from the sensor to the world frame, where they were compensated for the tool weight (CFINT subroutine).

---

[1] The forward kinematics required 50 floating point multiplies, 20 floating point adds, and 55 floating point memory calls. This includes calculation of the ${}^0R_6$ transformation matrix. If only the P_x and P_z position elements were required the number of calculations would be significantly less.

The 1.4 ms did not include the time to compute the ${}^0R_6$ transformation matrix because the forward kinematics were part of the basic control law.

The 0.25 ms of overhead time consisted of 0.10 ms to save test data, and 0.15 ms to perform diagnostics. The 0.10 ms required to save data accounted for the time to update the position and force data buffers used to save the test results. The save data operation involved writing 12 floating point values to memory and updating the buffer pointers. The 0.15 ms of diagnostics primarily consisted of writing integer values to predetermined locations in memory. These were useful for debugging the program but are not necessary for an operational controller.

5.2.2 *Sample Time Effects.* The effect of the sample and hold process used in the digital control system is the same as adding a transport lag or dead time to a continuous control system. Using Nyquist's stability theorem, D'Azzo and Houpis have shown increasing the dead time decreases the system stability (i.e. decreases damping) and can cause instability if the dead time is sufficiently large [7, pp. 295–296].

Another way to view the effect of sample time on system performance is to examine the nature of the techniques used to establish the desired dynamics in Section 3.1.2. These techniques replace the poles and zeroes of the uncontrolled robot dynamics with those of the desired dynamics. The desired dynamics are chosen using continuous time techniques (analog) in the s-plane, although the system actually uses a sampled-data digital controller and can only be exactly modeled in the discrete domain (z-plane). The use of continuous time techniques to analyze a discrete system is valid for small sample times[2]. If the sample time is small enough for the real coordinate of the s-plane poles (σ) to satisfy

---

[2] Houpis and Lamont have called this the Pseudo-Continuous Time technique. It requires a Padé approximation of the sample and hold process, and the use of bilinear transformations between the s-plane and z-plane [22, pp. 248–256].

$$\sigma \geq \frac{-0.1}{T_s} \tag{5.2}$$

then there will be a high correlation between the continuous and discrete time responses. If the s-plane poles are to the left of the σ bound, the s-plane model cannot accurately predict the response of the digitally controlled system. The desired damping ratio and natural frequency of the s-plane model will not be the same as the actual sampled-data system's response because of pole warping[3].

For an impedance controller, with $T_s = 0.014$ seconds, the limit for an accurate correlation of the continuous time model and discrete time response requires $\sigma \geq -7.14$. Values of K, M, and B yielding poles to the left of this limit (in the s-plane) will not be as well damped as intended. The $\sigma \geq -7.14$ limit restricts the natural frequency of the desired dynamics to undesirably low values. A smaller sample time would allow larger $|\sigma|$, and therefore higher natural frequencies for a given damping ratio. Using higher natural frequencies would allow the manipulator to respond to higher frequency inputs, and reduce the steady-state response error. With larger damping coefficients (σ) the duration of the system's transient response could be shortened, since settling time (t_s) is

$$t_s = \frac{4}{|\sigma|} \tag{5.3}$$

The end result of decreased sample time would be an improved response of the desired dynamics.

The natural frequency of the desired dynamics is also limited by the need to avoid exciting uncontrolled, high frequency, body bending vibrational modes of the PUMA link structure. However, if these vibration modes could be controlled by the impedance controller, than they could be prevented from resonating and destabilizing the PUMA arm. To control these modes Fu, Gonzalez and Lee suggest Nyquist's (or Shannon's) sampling theorem be modified to give [16, p. 222]

---

[3] Pole warping due to the bilinear Tustin transformation is described in detail in Chapter 7 of [22].

$$T_s = \frac{1}{20 f_n} = \frac{\pi}{10 \omega_n} \tag{5.4}$$

where f_n and ω_n are the frequencies in Hertz and radians per second, respectively, of the highest frequency vibrational modes excited by the controller. With the sample time of $T_s = 0.014$ seconds used in the tests of the impedance controller, the largest frequency which could possibly be controlled was 3.6 Hz.

Lastly, shorter sample times have an ancillary benefit. Because the control algorithm uses position information from the previous increment to calculate the $I(\vec{q})$ and $J(\vec{q})$ matrices of the impedance coefficients, these coefficients are always one interval behind the current arm position. As the sample time decreases, the change in $I(\vec{q})$ and $J(\vec{q})$ from one interval to the next becomes smaller. So with smaller sample times, less inaccuracy is introduced by using the $I(\vec{q})$ and $J(\vec{q})$ matrices based on the previous interval.

5.3 Trajectory Tracking.

Because the impedance control law includes a model of the manipulator dynamics, as well as position and velocity error feedback loops, it should perform well for freespace trajectory tracking. The ability of the manipulator to track three different unconstrained trajectories was evaluated. The trajectories were:

Stationary — the tool was commanded to remain stationary at X=-0.32565 meters, Y=-0.14909 meters, and Z=0.11439 meters. This corresponded to the joint position $\vec{q} = [0, -135, 135, 0, 0, 0]^T$ degrees.

Linear — the tool was commanded to descend in a line parallel to the world frame Z axis at constant velocity. The resulting cartesian positions were X=-0.93390 meters, Y=0.14909 meters, and Z=0.01853 to -0.33086 meters. The position at the start of the trajectory corresponded to a joint position of

$\vec{q} = [0, -115, 0, 0, 0, 0]^T$ degrees. A plot of the desired tool position in world coordinates is shown in Fig. 5.1.

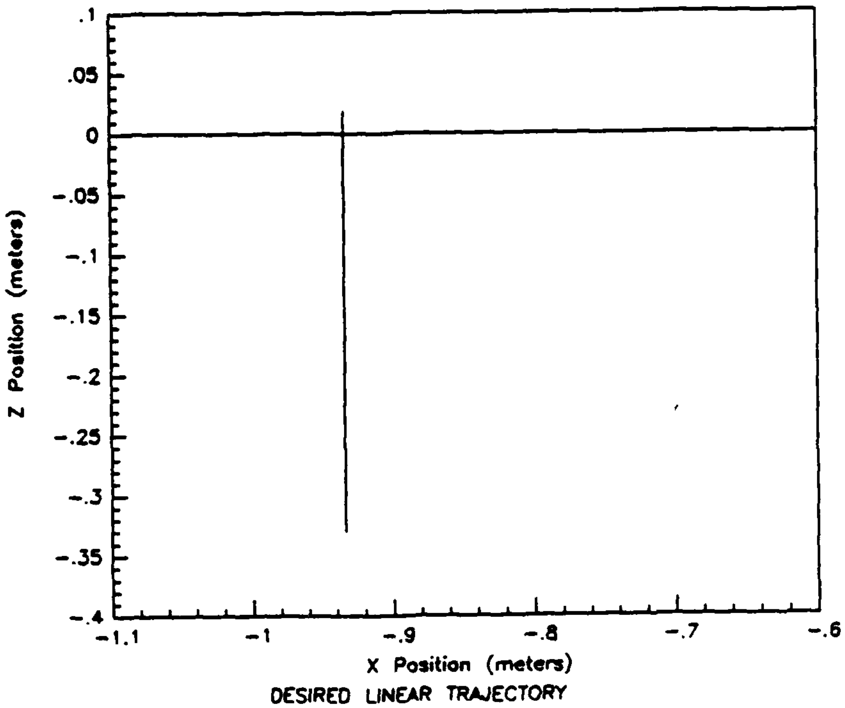
Circular — the tool was commanded to trace a circle 0.175 meters in radius centered on X=-0.934 meters, Y=0.14909 meters, and Z=-0.156 meters. The tool started at the top of the circle (X=-0.934 meters and Z=0.019 meters, joint

position $\vec{q} = [0, -115, 0, 0, 0, 0]^T$ degrees) and proceeded in a positive X direction. This traced a circle in a plane parallel to the world XZ plane. The circle was clockwise, as shown in Fig. 5.2.

The stationary trajectory tested the controller's ability to track a constant position trajectory having zero velocity. It was the simplest trajectory and indicated if the controller was stable. The linear trajectory provided the controller with a ramp input for the Z axis position, and a constant input for the X axis position (see Fig. 5.1). This trajectory required a constant velocity for both axes, except for the initial velocity jump along the Z axis. The circular trajectory required the end-effector position and velocity to follow sinusoidal trajectories in both the X and Z directions (see Fig. 5.2). This was the most complex trajectory, following the same shape as the virtual trajectory used by Hogan to test his impedance controller [20] (see Fig. 2.5).

5.3.1 *Results*. The results of the three trajectory tracking tests used to evaluate this initial implementation of the impedance controller are presented in Sections 5.3.1.1 through 5.3.1.3. The results are further discussed in Section 5.3.2 where three different mechanisms influencing the performance are described.

5.3.1.1 *Stationary Trajectory*. Figure 5.3 indicates typical performance of the impedance controller when tracking the stationary trajectory. Control parameters used in this test are shown in Table 5.4, where the values of the desired



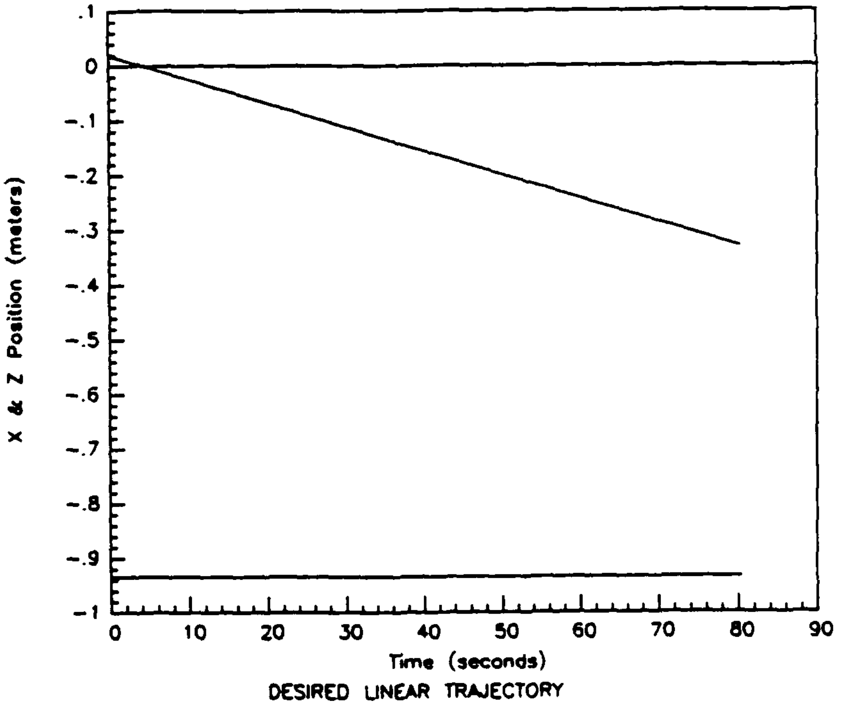


Figure 5.1. Desired Linear Trajectory. *Top:* Desired tool position in world X, Z coordinates. The Y coordinate is constant. *Bottom:* Desired X and Z position as a function of time (lower and upper curves, respectively).

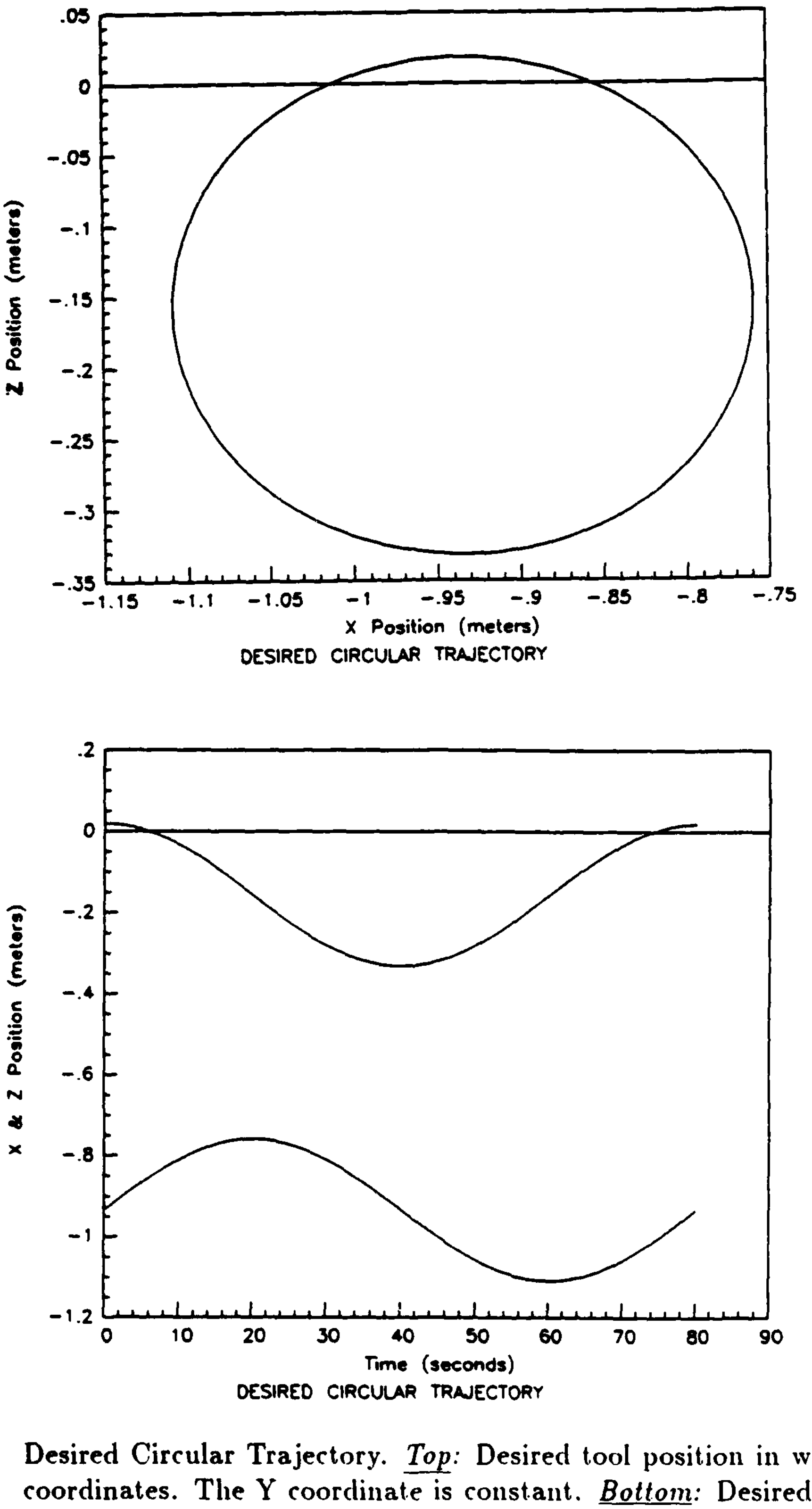


Figure 5.2. Desired Circular Trajectory. *Top:* Desired tool position in world X, Z coordinates. The Y coordinate is constant. *Bottom:* Desired X and Z position as a function of time (lower and upper curves, respectively).

Table 5.4. Stationary Trajectory Test, Controller Parameters

Parameter	*Value*
K_{11}, K_{22}	300 N/m
B_{11}, B_{22}	85.714 N-s/m
M_{11}, M_{22}	6.122 Kg
Sample Time	14 ms
Interpolations	4 per cycle
Deadband Threshold	0.0 N
ω_n	7.0 rad/s
ζ	1.0

dynamics natural frequency (ω_n) and damping ratio (ζ) were derived from the diagonal K, B, and M matrices using the model of the desired second order dynamics discussed in Section 3.1.2.

The impedance controller tracked the stationary trajectory quite well, except for a slight 1.5 cm jump in the X position midway through the trajectory. Because of the particular joint positions used in this test, movement in the X direction was possible without significant movement in the Z direction. This could occur through a slight rotation about joint 3; in this case the 1.5 cm jump represented a 1.3 degree rotation. The exact explanation for the jump is unclear. However, it could have occurred when high frequency vibration in the arm's gear train caused link 3 to suddenly break loose from the joint stiction effect and rotate. Because the error was small, the impedance controller did not command enough corrective torque to overcome the high stiction torques in the arm, and the error went uncorrected.

The initial ($t = 0$ s) 0.5 cm jump in the X and Z position of the arm was due to start-up transients as the impedance controller began to track the trajectory. Like the jump midway through the trajectory, this constant error was too small for the resulting corrective torque to overcome the stiction in the arm.

5.3.1.2 *Linear Trajectory.* Figure 5.4 shows the actual path in the XZ plane of the tool end point while tracking the linear trajectory. Figure 5.5

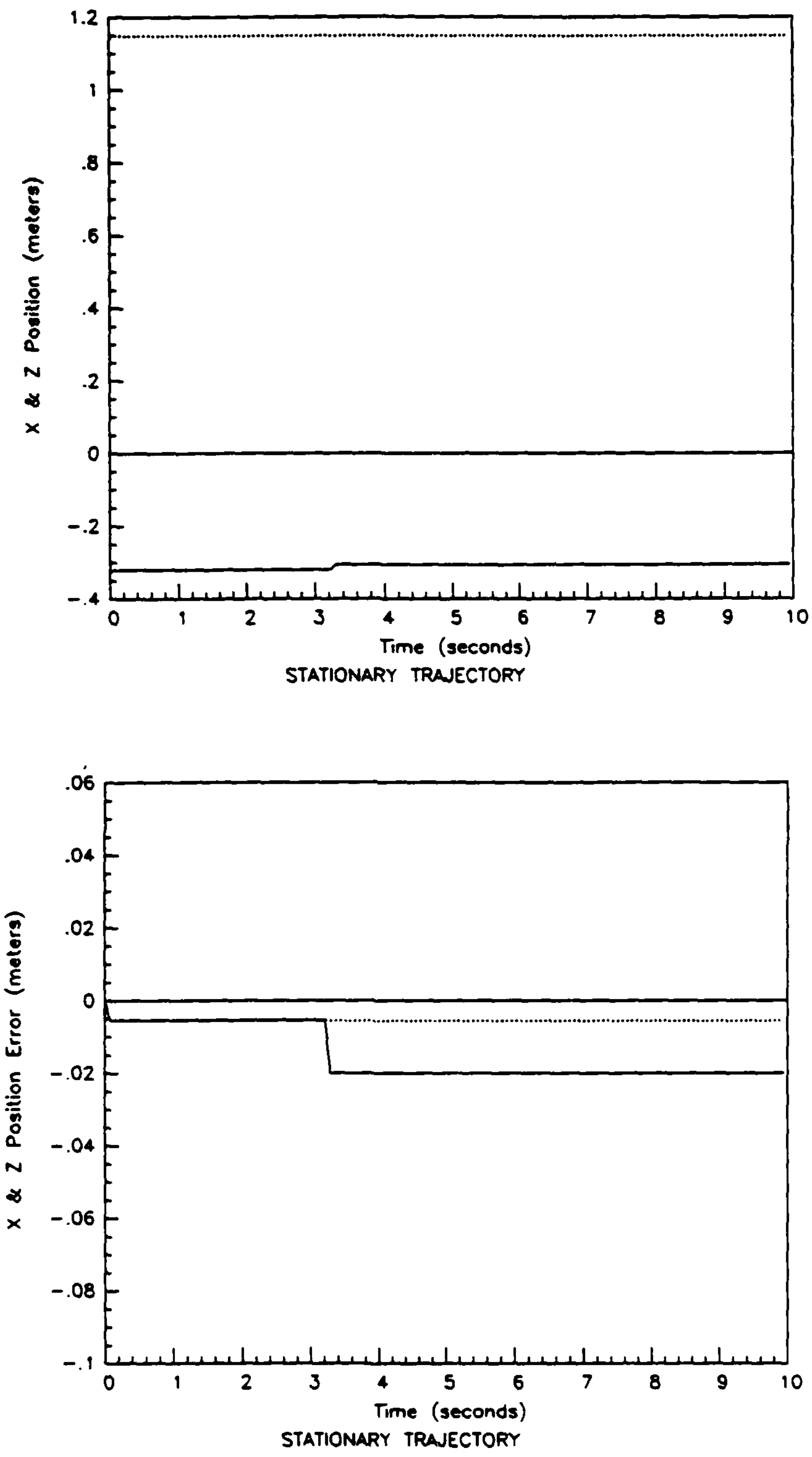


Figure 5.3. Stationary Trajectory Tracking Results (X=solid line, Z=dashed line). *Top:* The overall position of the tool in X, Z coordinates. *Bottom:* Error between the desired position and the actual position.

Table 5.5. Linear Trajectory Test, Controller Parameters

Parameter	*Value*
K_{11}, K_{22}	300 N/m
B_{11}, B_{22}	69.0 N-s/m
M_{11}, M_{22}	3.0 Kg
Sample Time	14 ms
Interpolations	19 per cycle
Deadband Threshold	±6.0 N
ω_n	10.0 rad/s
ζ	1.15

displays the actual position and position errors with respect to time. Table 5.5 lists the controller parameters used in this test.

The controller generally remained close to the linear trajectory, however both the X and Z positions were in error. Observing the manipulator, it was seen to move along the trajectory with an oscillatory motion, punctuated by apparently random pauses where the robot came to halt. At times the motions of links 2 and 3 did not appear to be fully coordinated, that is their motions were not sufficiently synchronized for the revolute manipulator to track the cartesian trajectory.

The X and Z position errors are shown at the bottom of Figure 5.5. The X axis error reached a maximum of 2.7 cm midway through the trajectory, with a 0.6 cm error at the end. The Z axis error was a maximum at -6.8 cm (i.e. 6.8 cm behind the desired position) midway through the trajectory, with a -2.8 cm error at the end. The errors in the X and Z axes were correlated, that is when the manipulator moved to correct errors in one axis it introduced errors in the other axis. This resulted from the previously noted uncoordinated motion.

The motionless pauses observed during the test can be seen in the X, Z position plot (upper plot) of Figure 5.5. These occurred intermittently from t=0 to 25 seconds, and are prominent again near t=35 seconds and t=46 seconds.

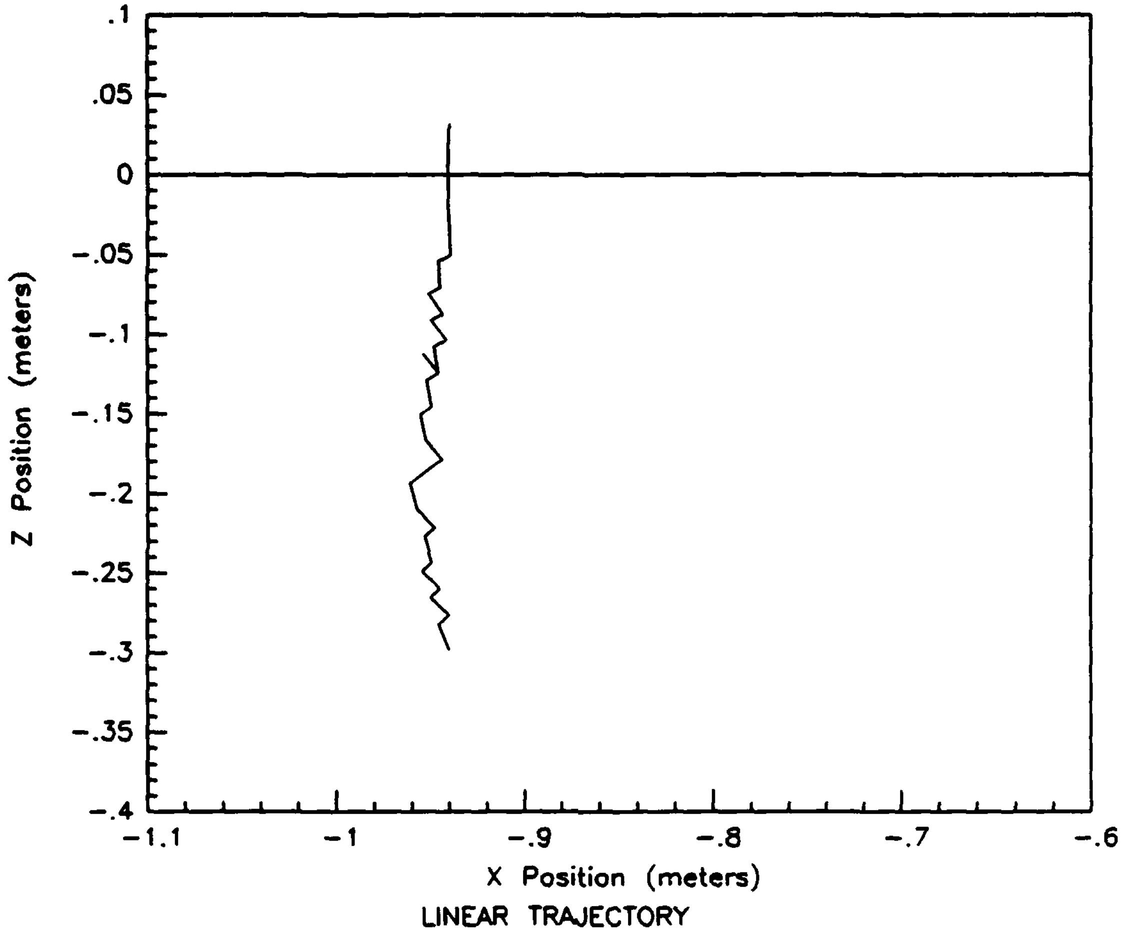
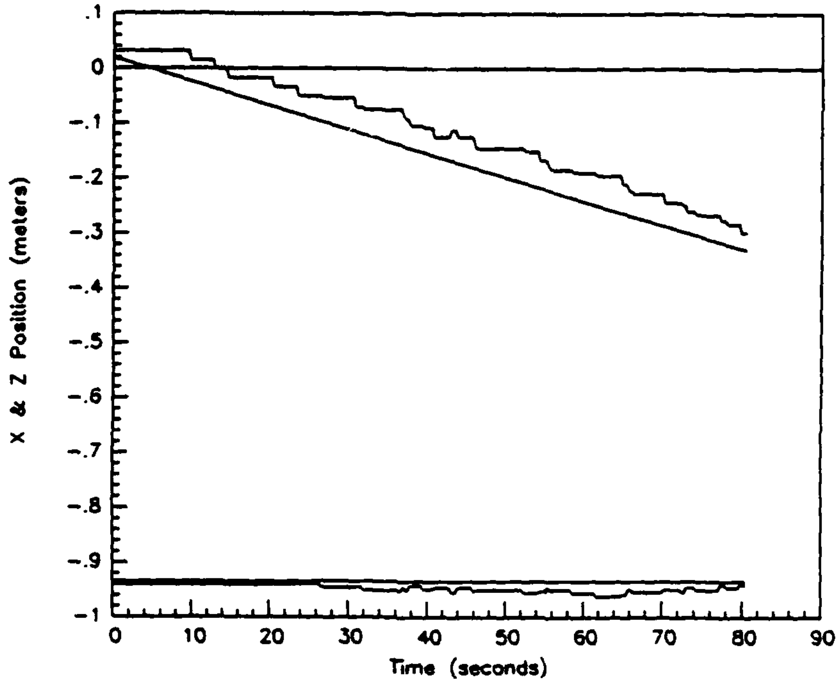
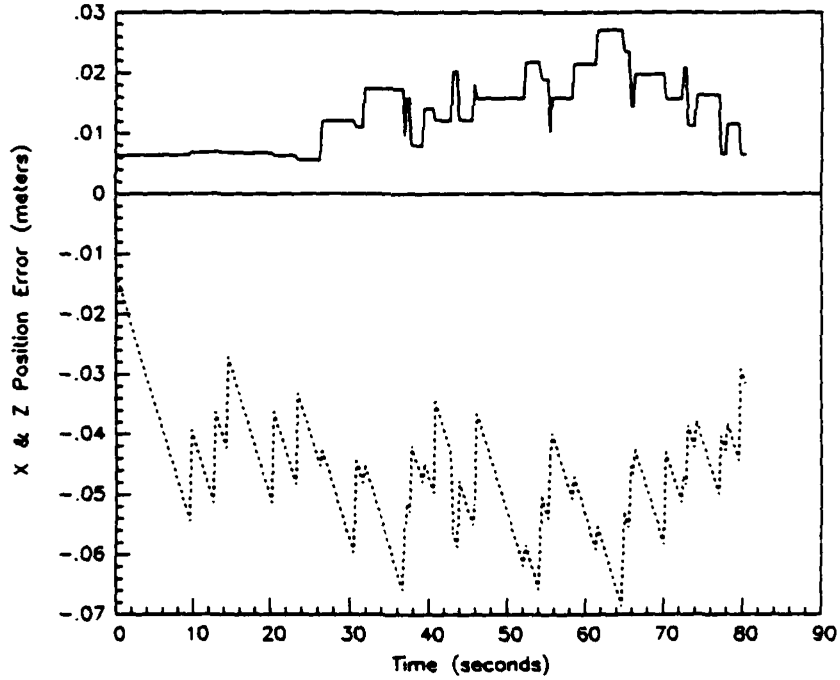


Figure 5.4. Typical Tool Position Tracking for the Linear Trajectory.



ACTUAL vs. DESIRED LINEAR TRAJECTORY



POSITION ERROR, LINEAR TRAJECTORY

Figure 5.5. Position and Position Error vs. Time, Linear Trajectory. *Top:* The impedance controller's ability to follow a ramp input in the Z direction is shown. A constant input is shown for the X direction. *Bottom:* Position error in the X (solid line) and Z (dashed line) directions.

Table 5.6. Circular Trajectory Test, Controller Parameters

Parameter	*Value*
K_{11}, K_{22}	300 N/m
B_{11}, B_{22}	85.714 N-s/m
M_{11}, M_{22}	6.122 Kg
Sample Time	14 ms
Interpolations	19 per cycle
Deadband Threshold	±6.0 N
ω_n	7.0 rad/s
ζ	1.0

5.3.1.3 *Circular Trajectory.* Figure 5.6 is a typical plot of the tool's actual versus desired position while following the circular trajectory. Figure 5.7 plots the tool position and position error as a function of time. Table 5.6 identifies the controller parameters used for this evaluation.

Observing the arm, it was seen to follow the circular trajectory in an oscillatory motion about the desired path. Frequently only one joint moved at a time, and occasionally the arm would come to a complete stop. These motionless periods can be seen in Figure 5.7 (upper plot) as points where both the X and Z positions are simultaneously constant. Examples occurred at t=18 seconds and t=51 seconds.

The error plot of Figure 5.7 (lower plot) indicates the tool's actual X and Z positions tended to lag behind the desired trajectory. This created a roughly circular track, displaced slightly to the left and above the desired path as shown in Figure 5.6. The amount of tracking error varied throughout the trajectory.

The large initial oscillations about the desired trajectory represent the transient response of the impedance controlled arm due to a sudden change in the arm velocity and an increasing position error. In an actual robotic application, these initial oscillations might be reduced through use of tailored trajectories having continuous position, velocity and acceleration values, and perhaps constraints on the amount of jerk (the time derivative of acceleration) imposed on the arm. The

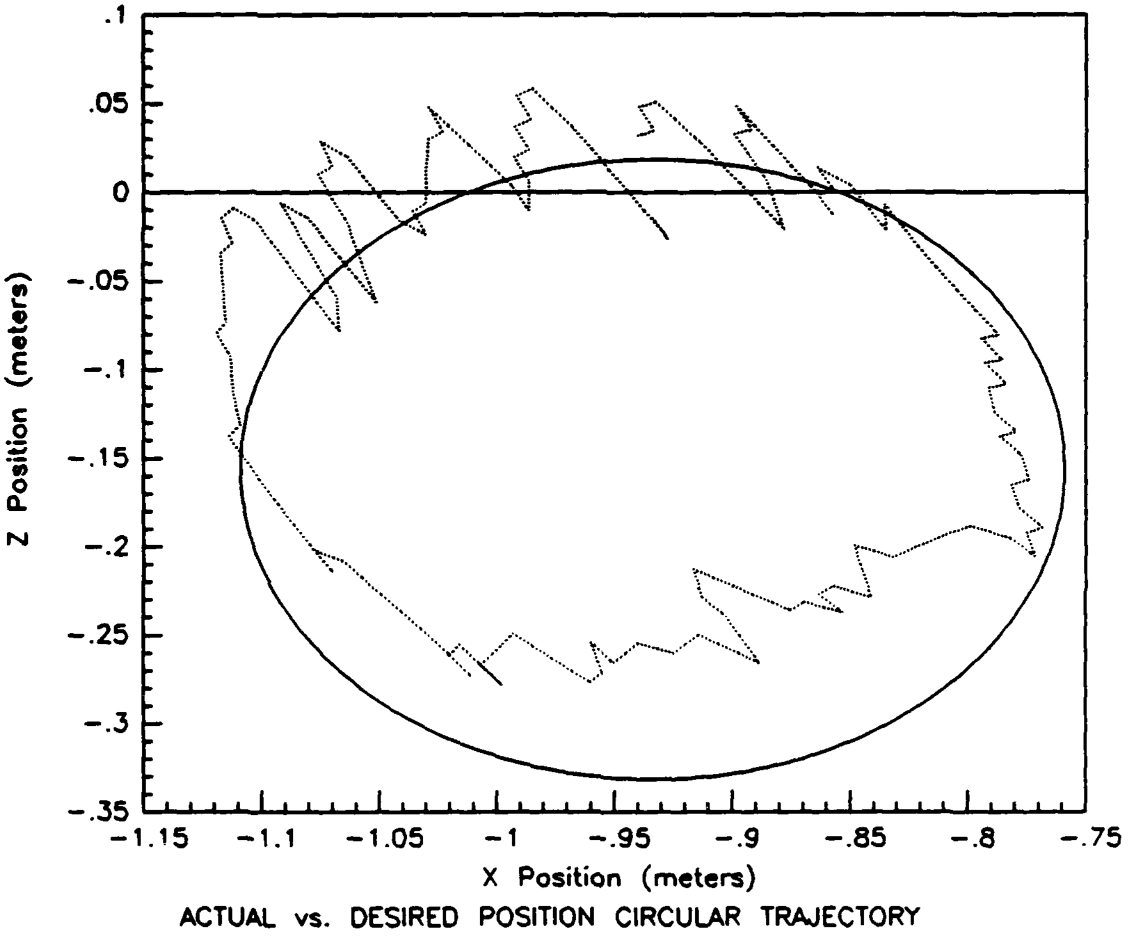
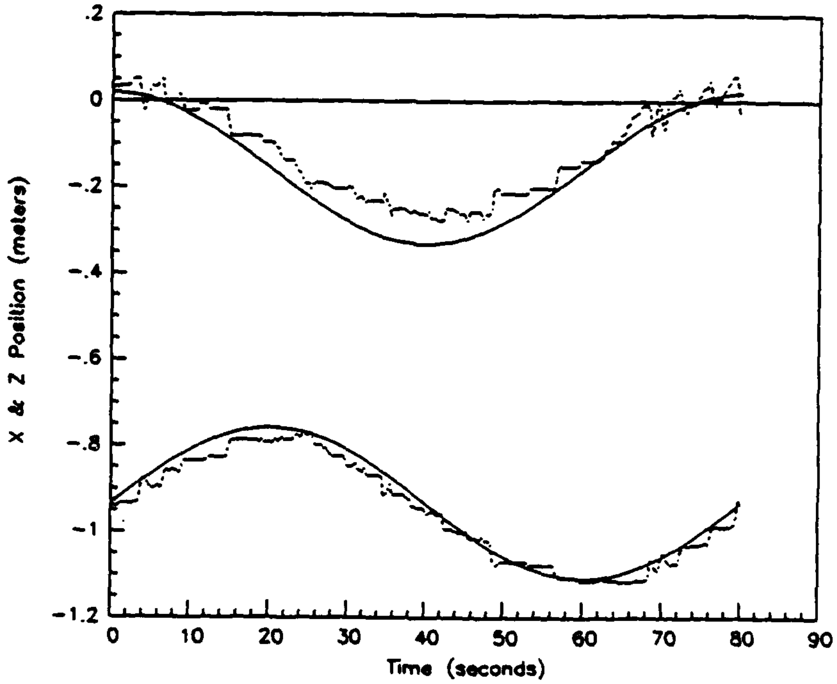
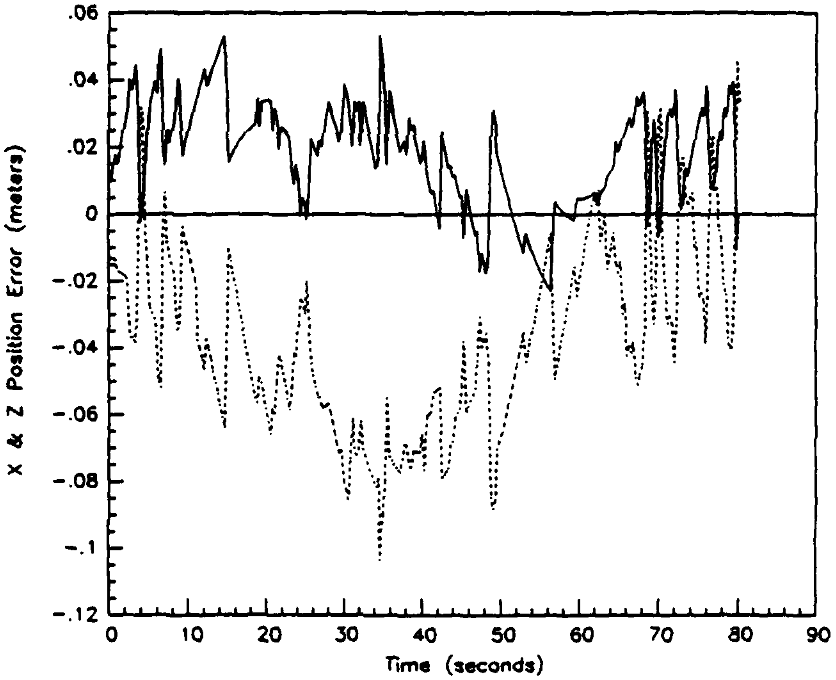


Figure 5.6. Typical Tool Position Tracking for the Circular Trajectory. The desired trajectory is the solid line and the actual path is the dashed line.



ACTUAL vs. DESIRED POSITION, CIRCULAR TRAJECTORY



POSITION ERROR, CIRCULAR TRAJECTORY

Figure 5.7. Position and Position Error vs. Time, Circular Trajectory. *Top:* Tracking a sinusoidal position curve for both X (lower curve) and Z (upper curve). Desired positions are shown with a solid line while actual positions are shown with dashed lines. *Bottom:* Position error in the X (solid line) and Z (dashed line) directions.

Z axis oscillation died out at the bottom of the circle, where the Z axis velocity is smallest, and then increased as the tool moved up the "side" of the circle, where the Z axis velocity is greatest. The Z axis oscillation behavior suggests a relationship between the controller's damping characteristics and the arm velocity. Oscillation in the X direction was roughly correlated with the oscillation in the Z direction.

5.3.2 Discussion of Trajectory Tracking Results.

The initial implementation of the impedance controller produced a system capable of rudimentary trajectory tracking. Performance of the controller was sufficient to allow preliminary testing of the compliant motion environment. While this initial impedance controller did not have adequate tracking performance for a full investigation of compliant motion control issues, the initial evaluations have identified the key areas requiring improvement. Three different effects appeared to be the most significant factors in degrading the performance of the impedance controller. They were:

- The degree of inaccuracy in the assumption of a commanded cartesian tool velocity equal to zero ($\vec{v}_o = 0$).
- The unmodeled stiction torque in the arm, and the uncertain accuracy of the Coulomb and viscous friction models, particularly at low speed.
- The long sample time required by the current controller implementation.

5.3.2.1 Velocity Effects.

In the development of the simplified impedance control law (Eqn. 3.21) from the full impedance control law (Eqn. 3.20), the commanded velocity of the manipulator was assumed to be close to zero. This assumption eliminated the Coriolis and centrifugal acceleration torques, but *most importantly* allowed the approximation $\vec{v}_o = 0$ in the $(\vec{v}_o - \vec{v})$ term of Eqn. 3.20. As a result, the performance of the control law becomes sensitive to the accuracy of the zero velocity approximation. To insure the motion of the arm was properly damped, the speed of the manipulator along the trajectory had to limited.

Table 5.7. Controller Parameters for Velocity Effects Tests.

Parameter	*Value*
K_{11}, K_{22}	300 N/m
B_{11}, B_{22}	69.0 N-s/m
M_{11}, M_{22}	3.0 Kg
Sample Time	14 ms
Deadband Threshold	±6.0 N
ω_n	10.0 rad/s
ζ	1.15

The impact of this approximation was easily demonstrated by evaluating the manipulator's tracking performance when it was tested several times using the same trajectory, but with different velocities for each test. The manipulator velocity was altered by changing the number of interpolations per set point (also referred to as interpolations "per cycle") used in tracking the trajectory. Since the time between each interpolation was fixed at the sample time (T_s), increasing the number of interpolations per set point increased the elapsed time per set point. Since for each test the trajectory's set points remained the same distance apart, the required manipulator velocity decreased when the number of interpolations increased. For example, a test made with 19 interpolations per set point had a velocity four times slower than a test made with four interpolations per set point (note: one cycle must be added to both the 19 and four interpolations to account for the cycle using the set point as the desired position).

Figure 5.8 shows the effect of a change in velocity on the X and Z position errors when the controller used the parameters of Table 5.7. The velocity was changed by running one test with four interpolations per set point, while the other test was run with 19 interpolations per set point. The tests were performed using the circular trajectory. In both the X and Z channels, the error is clearly greater at higher speeds. Figure 5.9 presents the X and Z position errors for a test using the controller parameters of Table 5.6, but with only four interpolations per set point instead of nineteen. This figure can be compared with the lower plot of Figure 5.7

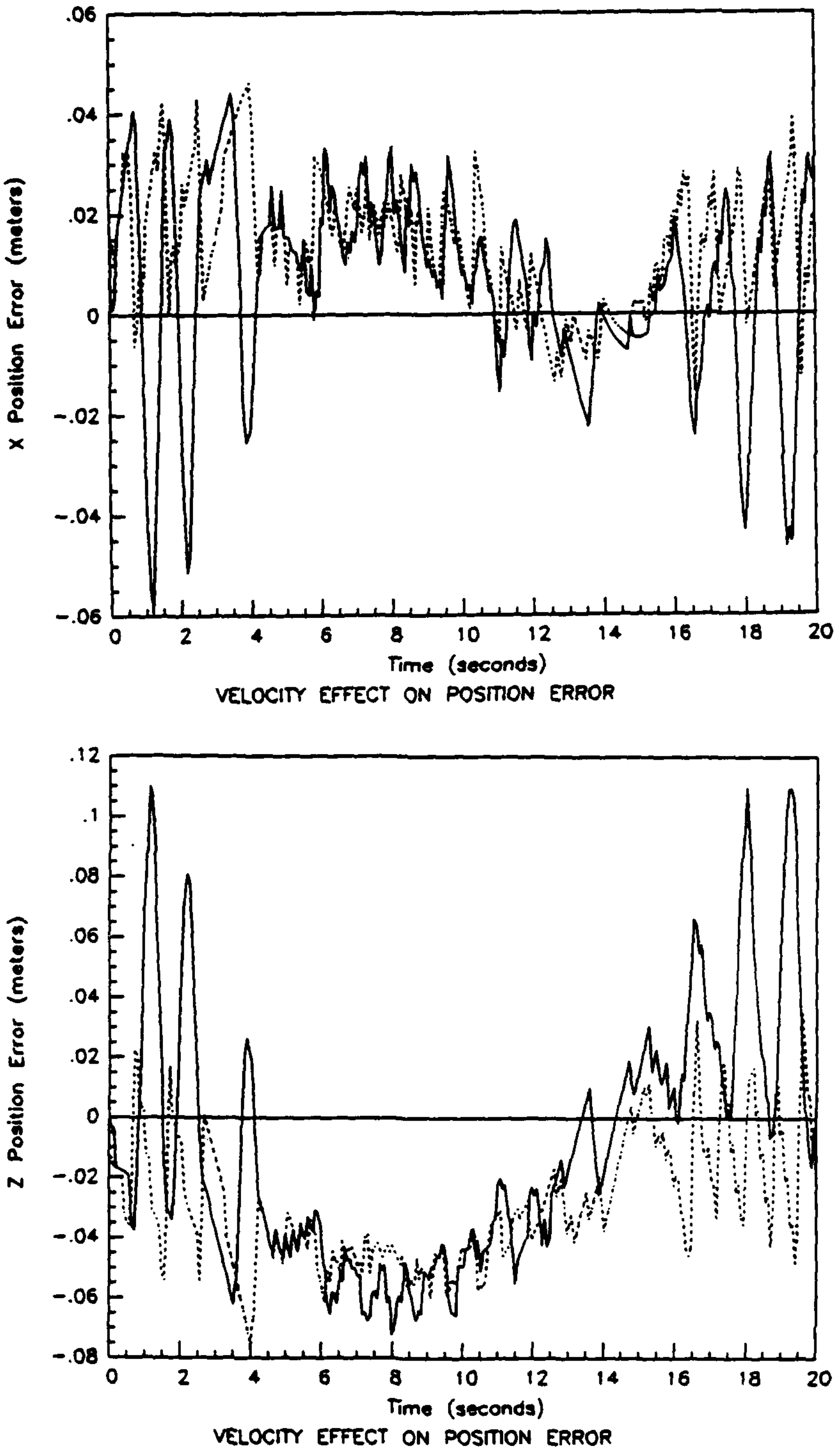


Figure 5.8. Velocity Effect on the X and Z Position Error. The solid line indicates error for four interpolations per set point. The dashed line indicates error with 19 interpolations per set point.

to again highlight the dependance of position error on velocity. In fact, the plot of Figure 5.9 represents an extreme case where the manipulator can be unstable for certain control parameter sets if the low velocity assumption is poor.

The ability of the simplified impedance controller to damp the arm motion improved as the velocity decreased. This can be understood by considering the impedance controller as a cartesian space PD control law whose gains are a function of the arm dynamic configuration (I^{-1} matrix). A reasonably accurate estimate of the velocity error, $(\vec{v}_o - \vec{v})$, is necessary for a PD control law to properly damp the motion of the robot. Incorrect damping results in unexpected oscillatory behavior, as shown in Figure 5.6, where even with a critically damped system ($\zeta = 1.0$), start-up transients and other oscillations occurred. Reducing the manipulator velocity improved the general performance of the simplified impedance control law, but did not produce entirely satisfactory results. Performance could be further improved by using an even slower velocity (which exaggerates the stiction problems discussed in Section 5.3.2.2), or by including the commanded velocity term in the control law. Fortunately, the compliant motion control environment has been set-up to allow the commanded velocity term to be included in the control law with several straight-forward modifications to the controller software (see Chapter Six).

5.3.2.2 *Friction Effects.* The simplified impedance control law includes a model for the Coulomb and viscous friction torques acting on the bearings, motors, and gear trains of each joint. The model was developed by Leahy based on empirical test data obtained from a PUMA 500 series robot [32]. The model can only be considered approximate because of the variance of the test data due to non-linearities such as backlash. Because it is an approximation, the model sometimes results in undercompensation of the joint friction torques. In addition, the friction model does not include any compensation for the stiction torques present at the joints when the joint velocities are zero.

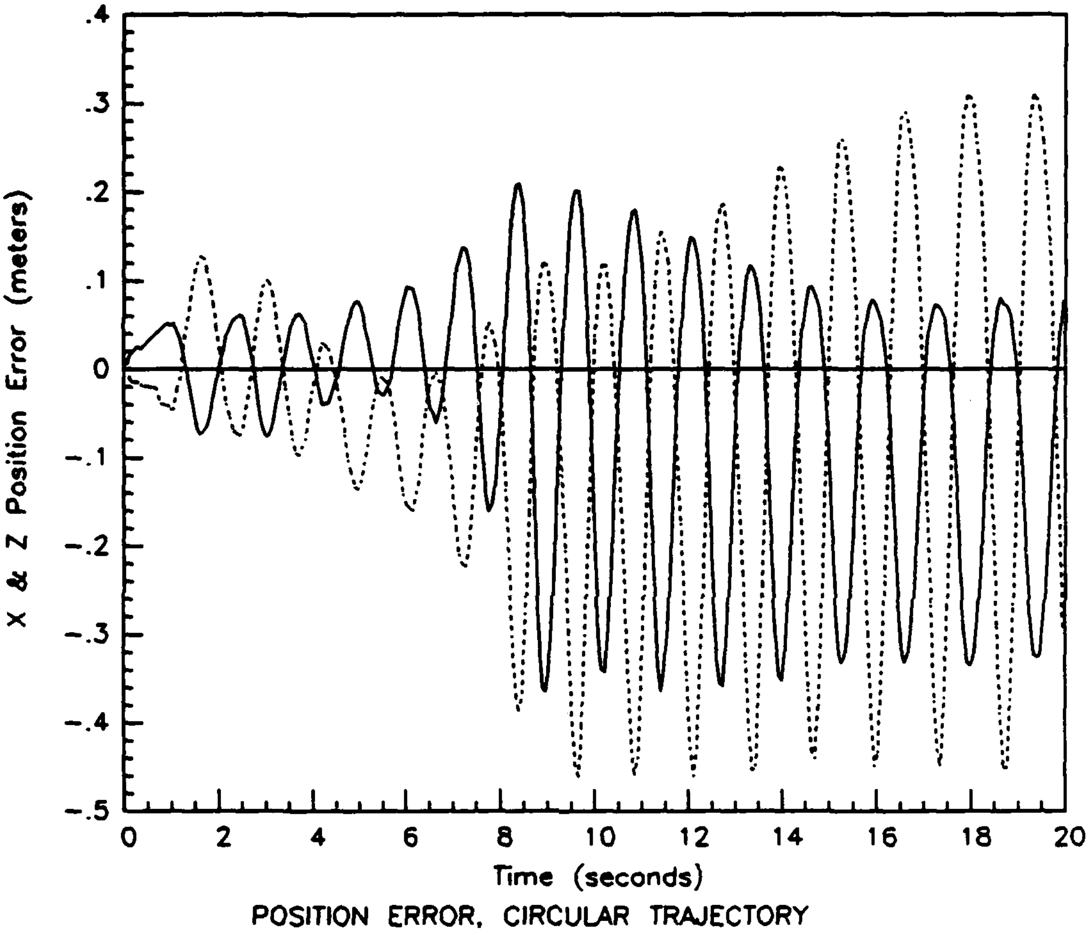


Figure 5.9. X and Z Position Error with Four Interpolations per Set Point. Table 5.6 lists the controller parameters used for this test (X=solid line, Z=dashed line).

At moderate to high speeds the friction model works well and is adequate for control of the PUMA [39]. At moderate speeds friction torques are significant, but not dominant when compared to the other dynamic effects (e.g. inertia, inertial coupling, Coriolis and centrifugal). Also, the greater the link momentum, the longer it takes for dissipative friction torques to absorb the link's energy and bring the link to a halt. So at moderate speeds the momentum of the three large links prevents the effects of insufficient friction compensation from becoming readily apparent. On the other hand, at the very low speeds used to insure proper damping with the impedance controller, friction is the *dominant* dynamic effect.

Inadequacies in the Coulomb and viscous friction model, and the lack of stiction compensation, account for several of the dynamic behaviors observed in the trajectory tracking evaluations. Approximations in the Coulomb and viscous friction model occasionally caused the friction compensation torques to be less than the actual friction present in the arm. When that condition occurred while the arm was moving slowly, friction was able to bring the arm to a halt. Once the arm was motionless, stiction dominated. Since the resistance due to stiction was greater than the Coulomb friction torque, the friction compensation torques were insufficient to alleviate the problem by themselves. The arm remained motionless until the position error became large enough for the feedback loop to generate actuator torques sufficient to overcome stiction. When the arm again began to move, the large position errors and sudden acceleration caused the arm to experience poorly damped transient behavior. As the transient oscillations were damped out, the arm slowed down, and the uncompensated friction could again stop the arm's motion.

When poor friction compensation had a greater impact on one joint than another, uncoordinated motion resulted. This occurred because the friction compensation torque sent to one joint was sufficient to overcome its internal friction, but the friction compensation torque sent to the other joint was not able to overcome that joint's friction. As a result, only one joint moved when the movement of both was required. In the trajectory tracking tests, joint 2 of the PUMA was more prone to the move-and-stall behavior than joint 3. The inability to simultaneously control the motion of both joints prevented the robot from making the coordinated movements necessary to precisely track the linear or circular cartesian trajectories.

In the stationary trajectory test (see Section 5.3.1.1) the controller was not able to correct for small displacement errors. Although part of this problem can be attributed to insufficient position error gain, stiction also played a role. Because of stiction, there was a minimum displacement error necessary to generate a large enough actuator torque to get the arm moving. If the errors fell below this threshold, they were not corrected, as in Figure 5.3. This same behavior occurred with the linear and circular trajectories.

Gear driven robots, like the PUMA, usually have noticeable friction forces involved in the gear train. Benton and Waters experienced problems when performing compliant motion experiments using a PUMA 560 arm because of friction induced non-linearities in the joint current-to-torque relationship [4, p. 70]. As noted in Section 2.3.1, Townsend and Salisbury found Coulomb friction and stiction could cause the manipulator to move in a "stick-slip" cycle very similar to the motion observed during the trajectory tracking tests [54, p. 886–887].

5.4 Compliant Motion Tests.

The compliant motion performance of the impedance controller was tested using the circular trajectory of Section 5.3 as the virtual path of the manipulator. The mockup refueling platform was placed in the path of the trajectory, and the test forced the manipulator to be compliant when it encountered the surface of the platform. Ideally, the compliant manipulator should cause the tool to smoothly slide across the constraint surface, while maintaining a minimum interface force. The tool position should follow a chord line across the lower portion of the circular trajectory, and the time history of the interface force should be directly proportional to the position error (see Fig. 2.5). This test was identical to the one used by Hogan to demonstrate his impedance controller on a simplified, two link, direct-drive arm (see Section 2.2.3). The controller parameters used in the compliant motion test were the same as those used for the previous circular trajectory tracking test (see Table 5.6). The test was run twice, first using active compliance from the force feedback terms in the impedance control law, and then using only the passive compliance inherent in the PD control loop for joint 5. The active and passive compliance cases were compared to demonstrate the benefits of adding active compliance.

5.4.1 Compliant Motion Test Using Active Compliance. The ability of the impedance controller to actively add compliance to the PUMA was tested using the trajectory and control parameters described above. The impedance controller successfully moved the tool through free-space and across the constraint surface. Excess interface force and unstable behavior were prevented from occurring. However, the performance of the robot could still be improved. During the test, the tool was observed to oscillate along the free-space portion of the trajectory. When the tool encountered the surface of the platform it had a tendency to bounce upwards by one to two centimeters. The tool moved across the platform with a series of low amplitude, low frequency oscillations. Contact with the platform remained continuous for only brief portions of the trajectory. As the desired trajectory cleared the platform, the tool again resumed an oscillatory path through free-space.

Figure 5.10 shows the actual tool path relative to the desired trajectory. The actual path was roughly semi-circular, and the oscillations created when the tool encountered the platform can be seen along the "straight" portion of the semicircle. As with the previous circular trajectory tracking test (see Section 5.3.1.3, Fig. 5.6), the tool motion was underdamped and there were significant transient response oscillations at the beginning of the trajectory. Figure 5.11 indicates the X and Z positions as a function of time. The constraint surface imposed a maximum error in the Z direction of 0.13 meters. This error in the Z direction created the interface force applied by the manipulator (based on the concept of stiffness, Eqn. 2.2).

The oscillatory, bouncing motion observed as the tool slid across the constraint surface was not entirely desirable, but was not unexpected. The desired dynamics established by the impedance control law are those of a mass-spring-damper system (see Fig. 3.4). Because the preliminary implementation of the impedance controller suffers from poor damping and an inability to make precision movements at slow speeds (as explained in Section 5.3.2), it was not able to control the oscillation of the spring mechanism and allow the tool to smoothly slide over the constraint surface. With improved damping and stiction compensation it should be possible to tune the M, B, and K parameter matrices to create the desired performance.

Figure 5.12 indicates the interface force in the X and Z directions. The deadband function was not applied to the interface force in order to improve the sensitivity of the controller to small forces. Eliminating the deadband also provided complete record of the interface forces. Without the deadband, the inaccuracies introduced into the force transformations (Eqn. 3.60) by the poor joint calibration created non-zero values of interface force along both the X and Z axes, even when the tool was not in contact with the surface. These bias errors were relatively small because the most poorly calibrated joints (joints 4 and 6) remained in the positions used to calibrate the force sensor. If joints 4 and 6 had been moved, the errors in interface force could have increased sufficiently to significantly influence the manipulator motion. In this later case use of the force deadband function would be necessary.

The high frequency oscillation of the interface force during the non-contact portion of the trajectory was the result of vibration in the tool. The tool vibration

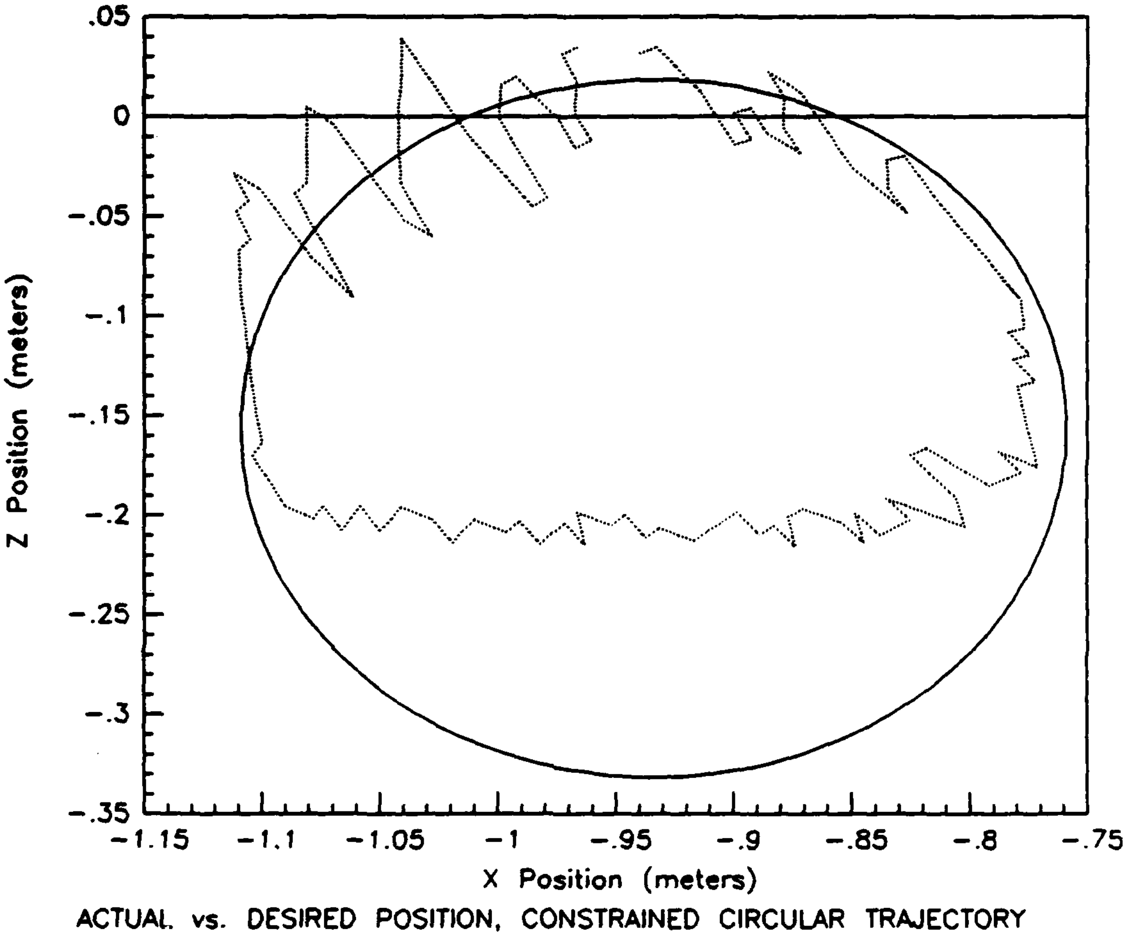


Figure 5.10. Tool Path for the Constrained Circular Trajectory Using Active Compliance. The virtual trajectory is shown as a solid line, while the actual path is a dashed line.

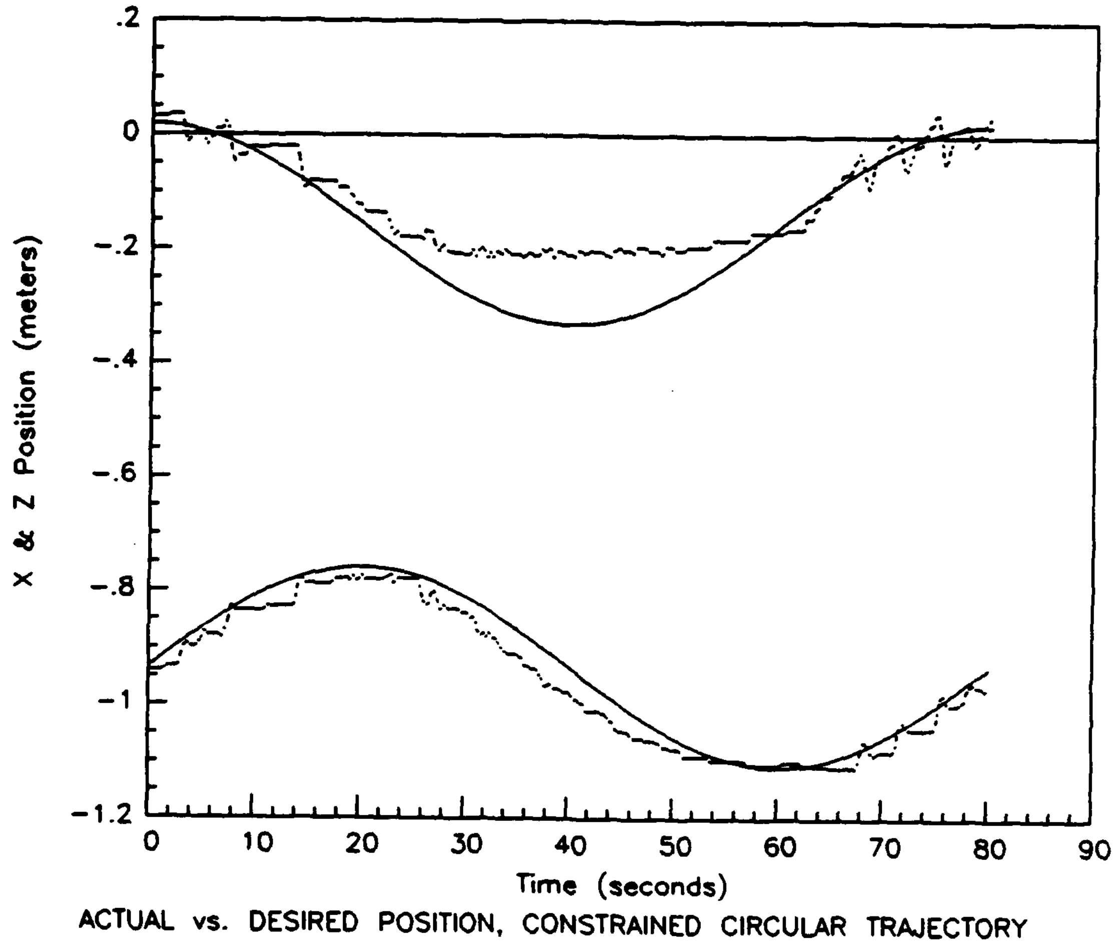
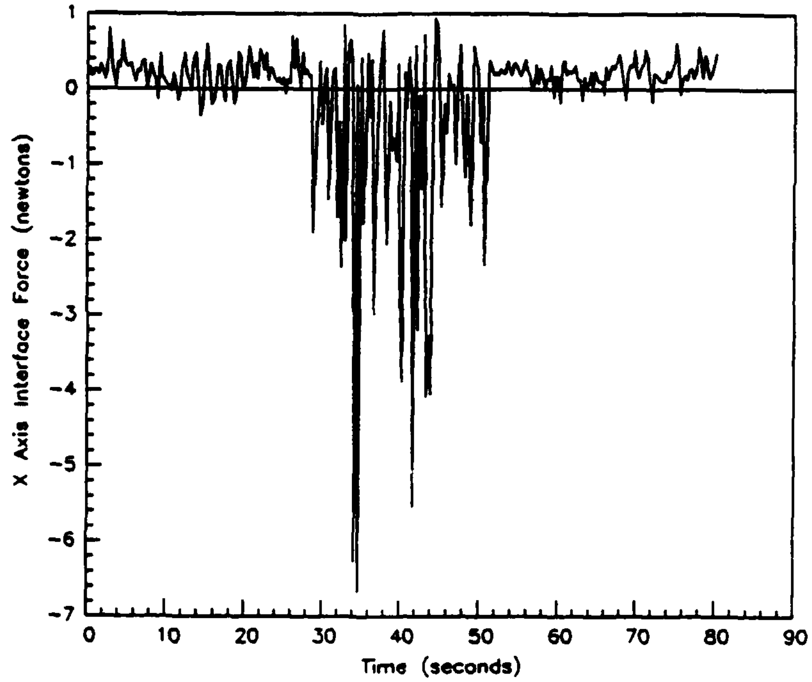
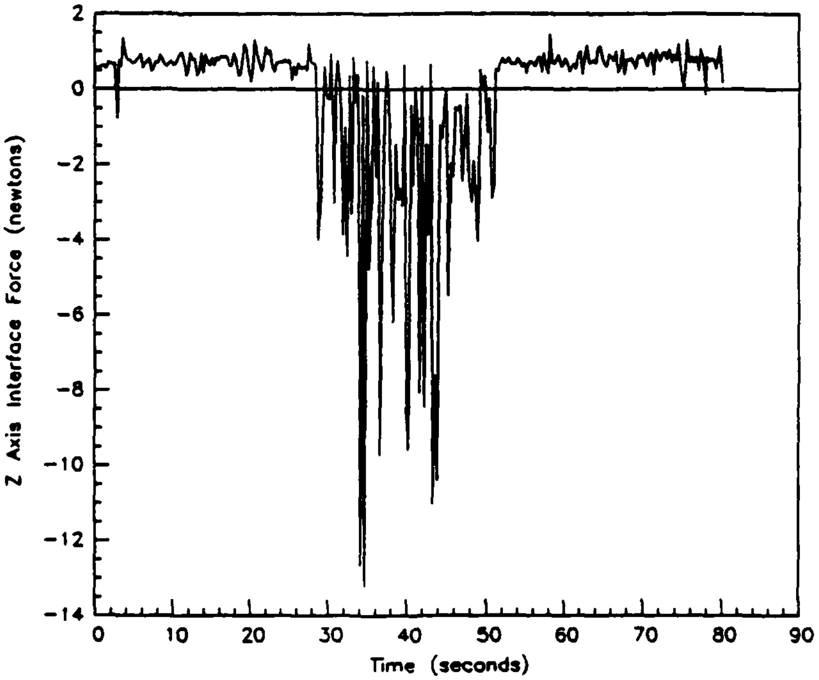


Figure 5.11. Tool Position with Respect to Time for the Constrained Circular Trajectory Using Active Compliance. The virtual trajectory is shown as a solid line, while the actual path is a dashed line. The lower curve is the X position while the upper curve is the Z position.



INTERFACE FORCE, CONSTRAINED CIRCULAR TRAJECTORY



INTERFACE FORCE, CONSTRAINED CIRCULAR TRAJECTORY

Figure 5.12. Interface Force for the Constrained Circular Trajectory Using Active Compliance. Note the difference in scale used for the X and Z forces.

was induced by the motion and vibration of the robot links. The wrist joints (joints 4, 5, and 6) are particularly prone to vibration since the gains of their PD control loop were not tuned for the mass of the tool and force transducer. Of the three wrist joints, joint 5 has the most effect on the tool vibration in the X and Z axes (assuming joints 1, 4, and 6 are kept near zero degrees).

The measured interface force in the Z direction (the lower plot of Fig. 5.12) represents the normal force between the tool and the platform surface. The applied force oscillated as the tool came in and out of contact with the surface. The instantaneous average value (a root mean square quantity) of the oscillating force data can be estimated as the center value of the oscillations. The instantaneous average value of interface force increased as the error between the virtual and actual paths increased, thus confirming the spring-like nature of the desired dynamics. The maximum value of the instantaneous average interface force was between two and three Newtons at approximately 36 to 40 seconds into the test. Peak forces of over 13 Newtons occurred at $t = 34$ seconds when an oscillation caused the tool to impact the surface particularly hard.

The measured interface force in the X direction represents the frictional forces encountered by the tool as it slid across the platform. As would be expected, the friction forces were correlated in time and magnitude with the normal forces. The correlation in magnitude was not as accurate as the correlation in time because scratches and nicks in the platform surface caused variations in the friction coefficient. In general the friction forces were about half the normal forces, suggesting a Coulomb friction coefficient of approximately 0.5.

5.4.2 Compliant Motion Test Using Passive Compliance. As discussed in Section 3.3, the positions of joints 1, 4, 5, and 6 were governed by a PD control law. Because the position error gains used in this control law were relatively low, they created an "electronic" remote center of compliance (RCC) using the four joints not governed by the impedance controller. The electronic RCC introduced passive compliance (i.e. compliance which is not explicitly controlled by feedback of interface force measurements with a feedback loop) into the PUMA arm. Passive compliance was demonstrated using the same test trajectory as was used to test the active compliance of the impedance controller. The controller gains were also the same as the ones used by the active compliance test (see Table 5.6). To eliminate the active compliance from the control law, the impedance control algorithm was altered to eliminate the force feedback terms of Eqn. 3.21.

Using passive compliance, the motion of the tool through free-space was the same as was observed in the previous trajectory tracking tests. When the tool contacted the platform it initially bounced several centimeters, but then settled on to the surface and did not lose contact until commanded to move off the platform by the virtual trajectory. However, this smooth motion was only possible because of the passive compliance present in joint 5. As the tool moved across the surface joint 5 made major deflections to accommodate the constraint force imposed by the robot.

Figure 5.13 shows the motion of the tool end-point as it slid across the platform. Figure 5.14 displays the X and Z position as a function of time. The motion across the surface (the chord line of the semi-circle) does not appear smooth in the figure because the tool position was calculated using the Servo processor forward kinematics routine, assuming joint 5 was rigidly locked in the $q_5 = 0$ position. Better data, indicating the exact tool position, was not available from the control system. In this figure the constraint surface appears to be located slightly higher than in Figure 5.10. The apparent change in height was due to differences in the PUMA joint calibration between tests.

Figure 5.15 traces the interface force in the X and Z directions throughout the course of the test. With passive compliance, the maximum instantaneous average value of interface force was estimated as 10 Newtons in the Z direction and four

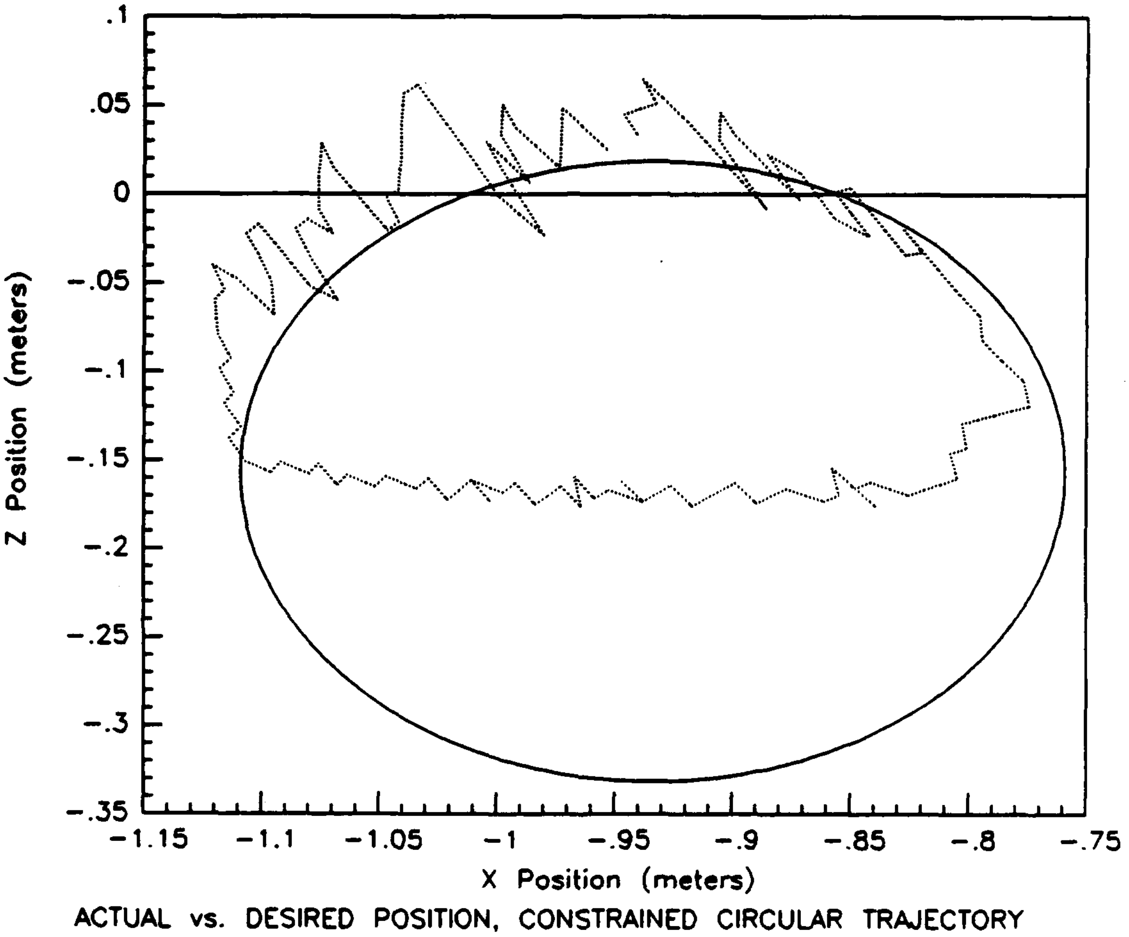


Figure 5.13. Tool Path for the Constrained Circular Trajectory Using Passive Compliance. The virtual trajectory is shown as a solid line, while the actual path is a dashed line.

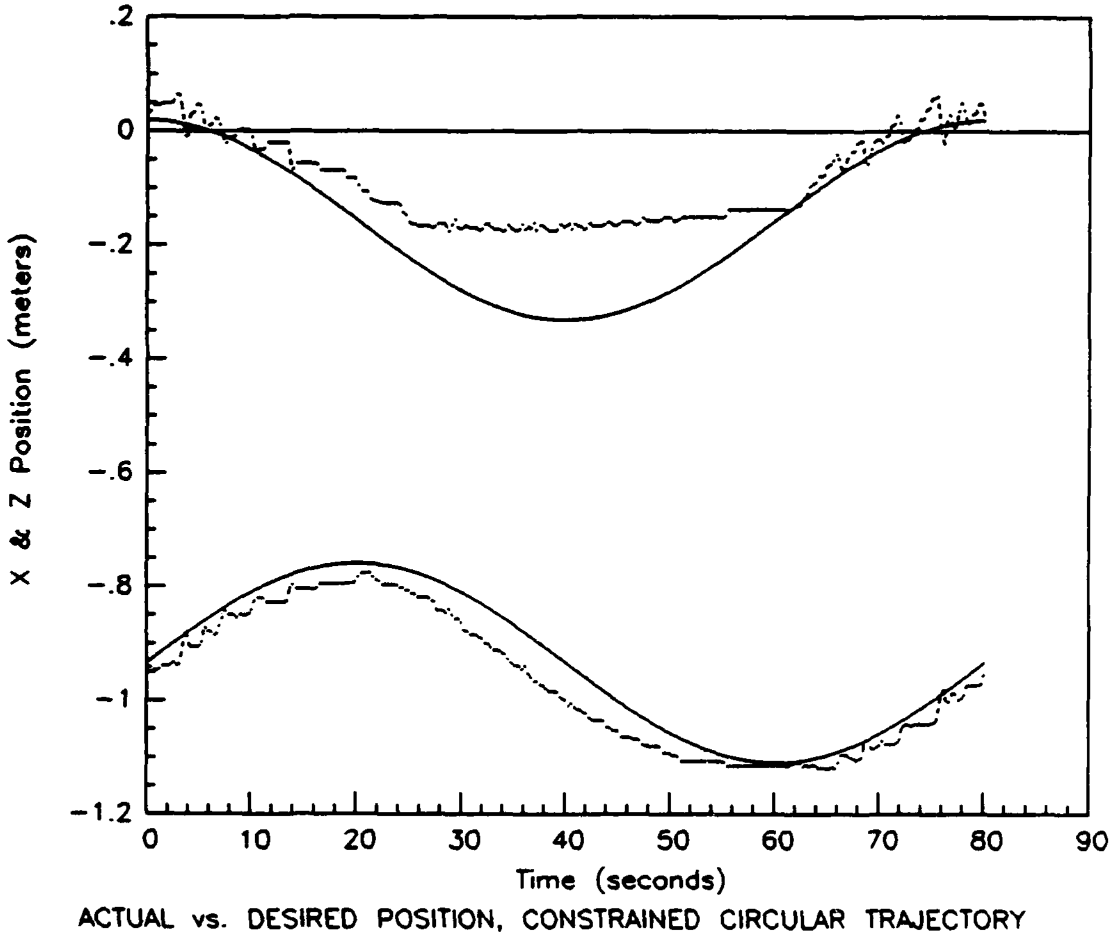
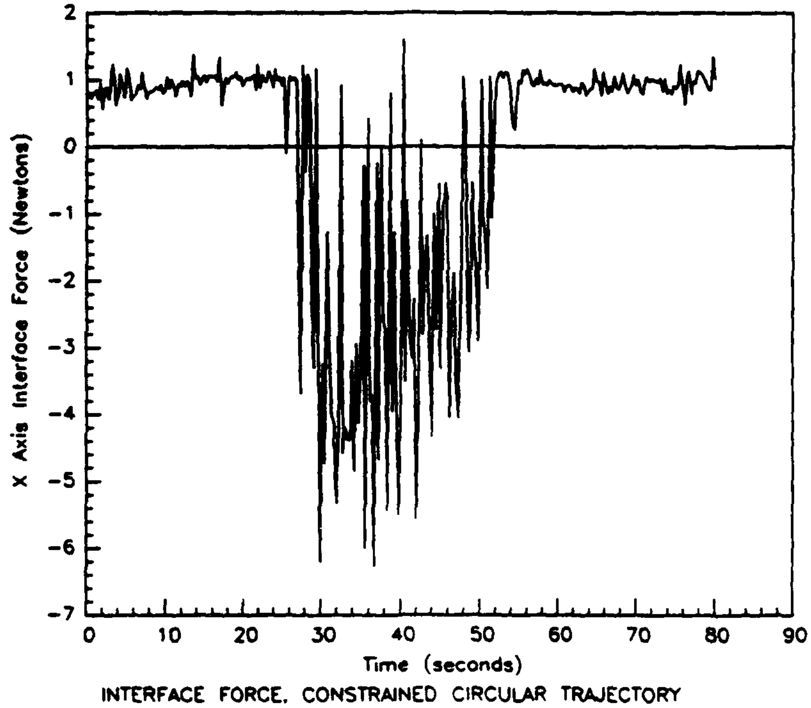


Figure 5.14. Tool Position with Respect to Time for the Constrained Circular Trajectory Using Passive Compliance. The virtual trajectory is shown as a solid line, while the actual path is a dashed line. The lower curve is the X position while the upper curve is the Z position.



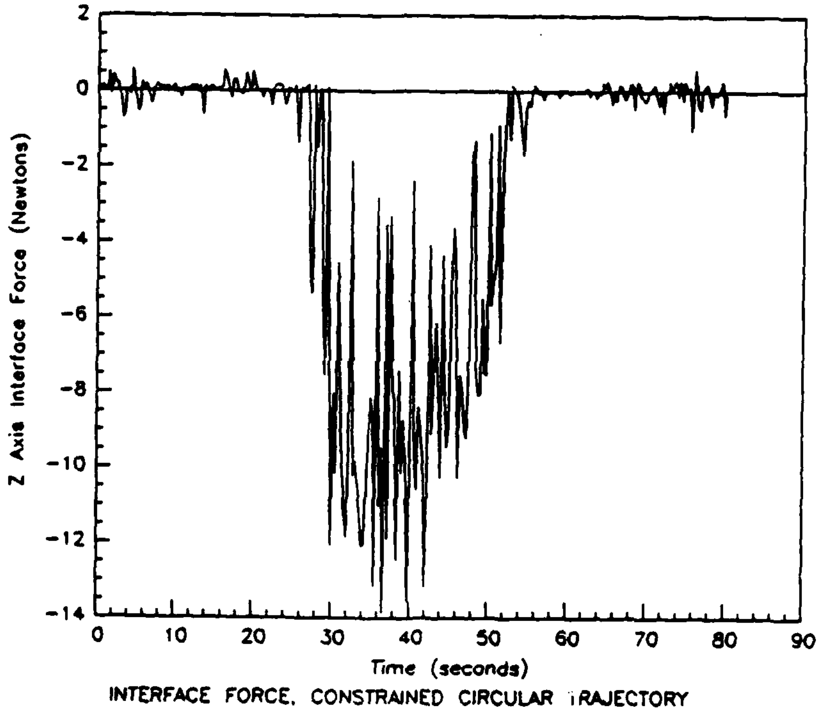


Figure 5.15. Interface Force for the Constrained Circular Trajectory Using Passive Compliance. Note the difference in scale used for the X and Z forces.

Newtons in the X direction. Using passive compliance, the instantaneous average values of the interface forces were approximately 300 percent higher than those generated using active compliance.

Some portion of this change can be attributed to changes in the joint position calibration. Calibration changes caused the tool position error to be approximately four centimeters greater for the passive compliance test than for the active compliance test. This represents a 30 percent increase in the position error relative to the 13 cm error of the active compliance test. Assuming a linear relationship between position error and force (an assumption justified by Eqn. 2.2), this 30 percent increase in error cannot account for the 300 percent increase in interface force between the two tests. *Clearly, the impedance controller is successfully using force feedback to reduce the interface forces and add active compliance to the robot manipulator!*

5.5 Discussion of Evaluation Results.

The simplified impedance controller implemented in this thesis has successfully demonstrated the use of active compliance to reduce the interface force between the manipulator and the environment. In addition, the performance of the force sensor and the execution time of the impedance control algorithm were quantified.

The force sensor performance is acceptable for use with the two degree of freedom impedance controller, since joints 4, 5, and 6 are not actually moved very far from their calibration positions. The absolute position calibration of the PUMA joints needs to be improved (especially joints 4 and 6) if the user desires to achieve a more accurate calculation of the interface forces. Alternatively, full six degree of freedom impedance control can be performed, without improving the arm calibration, by using deadband functions to eliminate errors in the force calculations. Unfortunately, these deadband functions will limit the sensitivity of the arm to small interface force values, and will introduce additional non-linearities into the control system.

The execution time of the impedance control algorithm's principal components were determined. Faster sampling was desired to improve the controller performance, however the use of the PUMA clock as the source of the controller timing signal limited sample times to multiples of 7 ms. The algorithm was executed within a 14 ms sample time since its measured execution time was approximately 9.7 ms (not including the time for diagnostic write statements). Therefore, if sample time improvements are to be realized, either the impedance control algorithm must be altered to execute in under 7 ms, or the compliant motion environment must incorporate a new source of controller timing signals allowing "finer" selection of sample times. With knowledge of the execution times, future researchers can better estimate the calculation time of compliant motion control algorithms, and these times can serve as the basis for a re-organization of the impedance control algorithm to shorten the sample time.

The trajectory tracking and compliance capabilities of the impedance controller were evaluated for several different trajectories. This initial implementation of the impedance controller successfully demonstrated rudimentary tracking capabilities. The tests pointed out two problem areas requiring improvements to obtain better trajectory tracking and compliant motion performance. These two problems were:

- The poor damping characteristics of the controller due to the zero velocity approximation and the relatively long 14 ms sample time.
- The significant impact on the arm dynamics of uncertainty in the Coulomb and viscous friction compensation model, coupled with the lack of stiction compensation.

Unfortunately, the simplest solutions to these two problems were in opposition to each other. The zero velocity approximation required the manipulator to move slowly to achieve the best damping (least oscillation). However, at low speeds underestimation of the Coulomb and viscous friction at the joints caused the manipulator to stall. Once the manipulator motion stopped, the significant, but unmodelled, stiction forces required large position errors to develop before motion resumed. These two problems resulted in oscillatory manipulator motion, punctuated by random pauses, and with poor coordination between the links. These problems inhibited precise manipulator tracking of cartesian trajectories. Solving these problems (by introducing a commanded velocity term, using a faster sample time, and improving the friction model) should lead to enhanced trajectory tracking and smoother compliant motion.

The performance of the impedance controller on the PUMA can be compared to the performance achieved by Hogan using essentially the same impedance control law on a dynamically less complex manipulator. Hogan was able to demonstrate better trajectory tracking with smoother compliant motion (see Fig. 2.6) because his manipulator did not exhibit the same low-speed stiction dominated behavior as the PUMA. He avoided excessive friction through the use of direct-drive links (eliminating sticky gear trains) and special low friction joints. As a result, he smoothly operated at low velocities, where the zero velocity approximation is valid. Hogan also used a 9 ms sampling time, thus allowing the use of "stiffer" controller gains (higher natural frequency pole placements) to improve controller performance without encountering the link resonant frequencies. The comparison of Hogan's manipulator to the PUMA points out key differences between the manipulators. Because of these differences it was possible for the same control law to be very successful on his manipulator, while it produced more limited results on the PUMA.

5.6 Summary.

In this chapter, the tests used to evaluate the compliant motion control environment and impedance control law were described. Sections 5.1 and 5.2 presented the results of the force sensor accuracy and algorithm timing tests. In light of these results, the impacts on controller performance of force sensing errors and long sample times were discussed.

The ability of the impedance control algorithm to control the PUMA during both trajectory tracking and compliant motion tests was evaluated. Section 5.3 indicated the tracking performance of the controller, and also discussed factors causing undesirable tracking behavior. Section 5.4 described how the impedance controller's ability to provide active compliance was successfully demonstrated. These initial tests highlighted the need to obtain improved trajectory tracking performance in future investigations of impedance control. A set of recommendations to improve performance are presented in the next chapter.

VI. *Conclusions and Recommendations*

The foundation has been laid for future compliant motion research in the AFIT Robotic Systems Laboratory. The compliant motion environment and prototype impedance controller developed by this research have successfully demonstrated elementary compliant motion. However, many improvements are possible and the recommendations made here should be pursued to fully exploit the possibilities of the compliant motion environment and the impedance controller.

6.1 Conclusions.

In this thesis the scope of work ranged from the design and fabrication of mockup refueling hardware, to the creation of a hierarchial control environment and the implementation of the impedance control law. With the bread-board compliant motion control environment it was possible to initially implement an impedance controller. The impedance controller successfully demonstrated elementary compliant motion, although the trajectory tracking performance requires improvement.

As a result of the effort to create the compliant motion environment, the following conclusions can be drawn:

1. A hierarchial compliant motion control environment now exists in the AFIT Robotic Systems Laboratory. The environment presents the user with a powerful, flexible interface for implementing compliant motion control laws. The compliant motion environment uses modular software and has a high degree of commonality to the existing ARCADE robot control environment. In general, this software supports six degree of freedom control of the PUMA robot.

2. Interface forces can now be determined in a fixed cartesian coordinate frame, and used for real-time control of the PUMA arm. This capability was made possible by integration into the environment of a wrist mounted, three axis force sensor measuring the force and moment applied about each of the sensor axes. An original technique was developed to transform these forces from the sensor to the world frames, while eliminating the weight of the tool from the force measurements.
3. A physical environment for testing compliant motion control techniques also exists. The environment includes half scale mockups of a standard USAF aerial refueling port and nozzle. The physical environment is reconfigurable and can support compliant motion tests ranging from simple surface following tasks to a demonstration of robotic aircraft refueling.
4. Based on the force sensor accuracy test results, it is apparent that the accuracy of the interface calculations can be significantly degraded by poor calibration of the PUMA arm (particularly joints 4 and 6). For applications where only joints 2 and 3 are moved, there are only minimal errors. For more general applications, either the calibration of the arm must be improved, or deadband functions must be added to prevent the arm motion from being influenced by force errors.

A simplified impedance control law was implemented in order to test the compliant motion control environment, and perform initial investigations of impedance control. This is the first known use of an impedance controller on a vertically articulated, gear driven, industrial manipulator. The impedance controller used links 2 and 3 of the PUMA arm to obtain two degree of freedom motion, however the implementation of the algorithm permits straight forward expansion to a more general six degree of freedom control law.

The impedance controller was tested using several simple free-space trajectories. The use of active versus passive compliance was evaluated using a simple surface following trajectory. The results of these initial investigations of impedance control demonstrate:

1. The impedance controller successfully adds active compliance to the manipulator system.
2. As currently implemented, the impedance control algorithm requires approximately 10 ms to execute. Unfortunately, force sensor sampling rates and the use of the PUMA clock limit the algorithm sample time to the undesirably long value of 14 ms.
3. Improved trajectory tracking is needed to provide smoother end-effector motion during both the free-space and constrained portions of compliant motion tasks. The current controller exhibits large position oscillations and uncoordinated joint motion during trajectory tracking tasks. Unacceptably large steady-state position errors exist and the manipulator moves in a "stick-slip" motion.
4. Poor trajectory tracking performance is the result of:
    - Inaccurately approximating the commanded velocity as $\vec{v}_o = 0$. The approximation prevents the control law from properly damping the end-effector motion and contributes to the undesired oscillations.
    - The use of an approximate model of the arm friction forces in the impedance control law. Underestimation of the Coulomb and viscous friction torques, and the lack of stiction compensation, results in the uncoordinated and "stick-slip" motions observed during the trajectory tracking tests. Stiction also prevents the arm from initiating motion to correct for small position errors. These friction problems are aggravated by the need to move slowly to satisfy the $\vec{v}_o = 0$ approximation.

- The restriction of the controller sample time to $T_s \geq 14$ ms. The 14 ms sample time restricts the locations of the desired dynamic's s-plane poles, resulting in poor damping and steady-state error. The relatively long sample time prevents the impedance controller from controlling the PUMA's high frequency body bending modes.

6.2 Recommendations.

The impedance controller implemented in this thesis requires three improvements to bring its trajectory tracking performance up to acceptable levels. Any one of the three improvements may result in dramatic performance improvements, so they should be implemented sequentially with the simplest improvements made first. The improvements are listed below, rank ordered from the easiest to implement to the most difficult.

1. The commanded velocity equal to zero assumption should be discarded and a commanded velocity ($\vec{v}_o$) term included in the control law. This will improve the controller's ability to damp the manipulator motion, and eliminate the need to restrict the manipulator to low speed trajectories.
2. An improved model of the friction torques present in the PUMA arm needs to be defined. A literature search should be initiated to develop a better understanding of friction modeling, as well as the motor and gear train effects producing the friction. A statistical study of the PUMA arm should be performed to re-evaluate the friction compensation model and insure the Coulomb and viscous friction torques are properly calculated. Also, the stiction forces acting at the joints should be estimated and added to the model.
3. The impedance control algorithm could be restructured to improve its calculation time. Faster calculation times would permit shorter sample times. With shorter sample times it should be possible to use control parameters

resulting in better manipulator damping and less steady-state error. The current impedance control algorithm calculation time could be reduced by using a separate processor to provide interface force information. Other changes to the Servo processor impedance control algorithm would be necessary to further reduce its calculation time. These additional changes could include adding a high speed processor to the MicroVax, and then using the MicroVax to perform the majority of the control law calculations now performed by the Servo processor. The Servo processor would then remain only as the interface to the force processor and the PUMA hardware level.

Once the impedance controller's trajectory tracking performance has been improved, it will be possible to tune the K, M, and B parameter matrices to produce the desired manipulator performance. At this point, it should be possible to perform a demonstration of robotic refueling using the two degree of freedom impedance controller. The control algorithm could also be expanded to allow six degree of freedom compliant motion. Then, when integrated with a machine vision system providing trajectory information, it will be possible to perform a demonstration of autonomous aircraft refueling.

Appendix A. *Control Law Derivation*

This appendix elaborates on the derivation of the full impedance control law. The derivation is based on the pole placement technique of algebraically comparing the desired dynamic response to the actual dynamic model. The resulting matrix equations can be solved for the actuator input values. The derivation presented here parallels the derivation presented by Hogan in Appendix A of [18]. Additional mathematical details have been added to provide a more complete understanding of the derivation.

From Eqn. 3.9 the desired dynamics are[1]

$$F_{int} = K(x_o - x) + B(v_o - v) - M\frac{dv}{dt} \tag{A.1}$$

solving for acceleration gives

$$\frac{dv}{dt} = M^{-1}K(x_o - x) + M^{-1}B(v_o - v) - M^{-1}F_{int} \tag{A.2}$$

Also, the manipulator dynamics are assumed to be (from Eqn. 3.19)

$$I\frac{d\dot{q}}{dt} + C(q,\dot{q}) + V(\dot{q}) + S(q) = \tau_{act} + \tau_{int} \tag{A.3}$$

The relationship between cartesian and joint velocities is given as [16, p. 544]

$$v = J\dot{q} \tag{A.4}$$

Differentiating with respect to time using the product rule gives

$$\frac{dv}{dt} = J\frac{d\dot{q}}{dt} + \frac{dJ}{dt}\dot{q} \tag{A.5}$$

Now, J is a function of $q(t)$, so $J = J(q(t))$ and differentiation of J with respect to time requires the chain rule.

$$\frac{dJ}{dt} = \frac{dJ}{dq}\frac{dq}{dt} \tag{A.6}$$

---

[1] Arrows indicating vector quantities have been removed

Differentiation of the matrix J with respect to the vector q produces a third order tensor composed of the gradients of the elements of J, then

$$\frac{dJ}{dq}\frac{dq}{dt} = \frac{dJ}{dq}(\dot{q}) = \begin{bmatrix} \dot{q}\cdot\nabla J_{11} & \dot{q}\cdot\nabla J_{12} \\ \dot{q}\cdot\nabla J_{21} & \dot{q}\cdot\nabla J_{22} \end{bmatrix} \qquad (A.7)$$

where $\frac{dJ}{dq}(\dot{q})$ is a tensor operator acting on $\dot{q}$, and $(\cdot)$ is the dot product. By substituting appropriately, Eqn. A.5 becomes

$$\frac{dv}{dt} = J\frac{d\dot{q}}{dt} + \frac{dJ}{dq}(\dot{q})\dot{q} \qquad (A.8)$$

Let $G(q,\dot{q}) = \frac{dJ}{dq}(\dot{q})\dot{q}$, the term is dependant only on the position and velocity of the joints, just like the Coriolis and centrifugal torque term $C(q,\dot{q})$. It acts to correct the Coriolis and centrifugal torques computed in the joint space for the cartesian nature of the control law.

Substitute $G(q,\dot{q}) = \frac{dJ}{dq}(\dot{q})\dot{q}$ in Eqn. A.8 and solve for $\frac{d\dot{q}}{dt}$

$$\frac{d\dot{q}}{dt} = J^{-1}\left[\frac{dv}{dt} - G(q,\dot{q})\right] \qquad (A.9)$$

Substituting into Eqn. A.3 yields

$$\tau_{act} + \tau_{int} = IJ^{-1}\left[\frac{dv}{dt} - G(q,\dot{q})\right] + C(q,\dot{q}) + V(\dot{q}) + S(q) \qquad (A.10)$$

Substitute for $\frac{dv}{dt}$ with Eqn. A.2

$$\begin{aligned}\tau_{act} + \tau_{int} &= IJ^{-1}\left[M^{-1}K(x_o - x) + M^{-1}B(v_o - v) - M^{-1}F_{int} - G(q,\dot{q})\right] \\ &\quad + C(q,\dot{q}) + V(\dot{q}) + S(q) \qquad (A.11)\end{aligned}$$

The cartesian position is determined by the forward kinematics function,

$$x = L(q) \qquad (A.12)$$

and from before $v = J\dot{q}$ so then

$$\begin{aligned}\tau_{act} + \tau_{int} &= IJ^{-1}M^{-1}K\left[x_o - L(q)\right] + IJ^{-1}M^{-1}B\left[v_o - J\dot{q}\right] - IJ^{-1}M^{-1}F_{int} \\ &\quad IJ^{-1}G(q,\dot{q}) + C(q,\dot{q}) + V(\dot{q}) + S(q) \qquad (A.13)\end{aligned}$$

To complete the derivation τ_{int} must be expressed in terms of F_{int}, since F_{int} is measureable using a wrist mounted force sensor. To make this replacement the relationship between joint torques and end-effector forces must be understood. The definition of virtual work gives

$$\delta W = F^T \delta x = \tau^T \delta q \tag{A.14}$$

then

$$F^T \delta x = \tau^T \delta q \tag{A.15}$$

again $\delta x = J \delta q$ so substituting

$$F^T J \delta q = \tau^T \delta q \tag{A.16}$$

eliminating the δq from both sides

$$F^T J = \tau^T \tag{A.17}$$

take the transpose of both sides

$$J^T F = \tau \tag{A.18}$$

so then the desired relationship is

$$\tau_{int} = J^T F_{int} \tag{A.19}$$

Returning to Eqn. A.13, substituting Eqn. A.19, and solving for τ_{act} completes the derivation of the full impedance control law

$$\begin{aligned} \tau_{act} = {} & I(q) J^{-1} M^{-1} K \left[x_o - L(q) \right] + S(q) \\ & + I(q) J^{-1} M^{-1} B \left[v_o - J \dot{q} \right] + V(\dot{q}) \\ & - \left[J^T + I(q) J^{-1} M^{-1} \right] F_{int} \\ & - I(q) J^{-1} G(q, \dot{q}) + C(q, \dot{q}) \end{aligned} \tag{A.20}$$

The equation has been arranged so the first row contains the position dependant terms, the second row the velocity dependant terms, the third row the force feedback terms, and the last row has the Coriolis and centrifugal terms.

Appendix B. *Aircraft Robotic Refueling Background*

B.1 Concept Overview.

Air Force interest in the use of robots to refuel aircraft stems from a desire to increase the rate of sortie generation and minimize the exposure of ground crews to hostile environments. Using an autonomous robot for refueling could decrease the number of man-hours required for refueling aircraft, thus decreasing the number of people required. Sortie generation rates can be increased by using these people to work on other aircraft, or other turnaround tasks. Alternately, in a nuclear or chemical warfare environment the number of casualties among ground crews can be reduced by allowing these people to remain in the safety of their protective shelters. Robotic refueling can be combined with other concepts for robotic aircraft ordnance loading and maintenance, to create a highly automated turnaround system. Such a system could have significant impacts on the time, manpower, and risks associated with aircraft turnaround.

The existing procedure for refueling aircraft is described by Battelle in [3, pp. 49–50, 116–117] and is summarized here. Currently aircraft are refueled on the ground through the single point refueling port, usually located underneath the wing or fuselage. A ground crew member maneuvers the fuel truck to the aircraft, and if necessary connects the truck's fuel pump with a flight line fuel hydrant. Next, he insures the aircraft and fuel truck are properly grounded. To access the single point refueling port a small accesss panel in the aircraft must be opened, usually by loosening several fasteners. The nozzle of the fuel line from the fuel truck is connected with the refueling port and the transfer of fuel is started. Fuel is transferred at a pressure of up to 55 psig. While the fuel is being loaded, the ground crew member checks the fuel tank air vents to insure air is flowing out of the tanks, preventing a dangerous buildup of pressure. Also, while the tanks are filling the ground crew member must visually inspect the aircraft and refueling equipment for leakage. When the job is completed the equipment is disconnected in reverse order. If a nuclear or chemical warfare environment exists, the crew member must perform the refueling while wearing MOPP IV protective mask, gloves, and coveralls [3, p. 11]. This protective gear impairs his performance, slowing down completion of the task and reducing his endurance.

In June 1986, the Air Force Flight Dynamics Laboratory contracted with Battelle Corporation for a study of robot applications to aircraft turnaround. Battelle examined the refueling task and conceptually defined a robotic refueler [3, pp. 26–32]. The concept developed by Battelle envisions a six degree of freedom arm mounted on a self-propelled cart. Fuel would be transfered through the aircraft's aerial refueling port instead of the single point refueling port (see Figure B.1). A refueling truck would tow the robot refueler cart to a point near the aircraft. After separating from the truck, the cart would be positioned near the front of the aircraft, with the refueling port within the workspace of the manipulator arm. In Battelle's technology limited (pre-1995 deployment) concept, a crewman would use tele-robotic techniques to insert the nozzle in the refueling port [3, pp. 11–12]. In more advanced concepts a vision system would locate the port and the manipulator arm could then insert the nozzle. A pre-determined amount of fuel could be transferred, or the tanks may be loaded until aircraft fuel sensors indicate the tanks are full. Vision systems would be used to monitor for leaks [3, p. 30].

Use of the aerial refueling port is desirable because:

1. For most aircraft a slipway exists to guide the nozzle to the port; exceptions are some older aircraft (e.g. F-4) where no slipway exists [25].
2. The aerial refueling port permits much higher fuel flow rates (up to 1200 gal/min) than the single point refueling port (up to 600 gal/min) [25]. Therefore, the time to refuel the aircraft is significantly reduced.

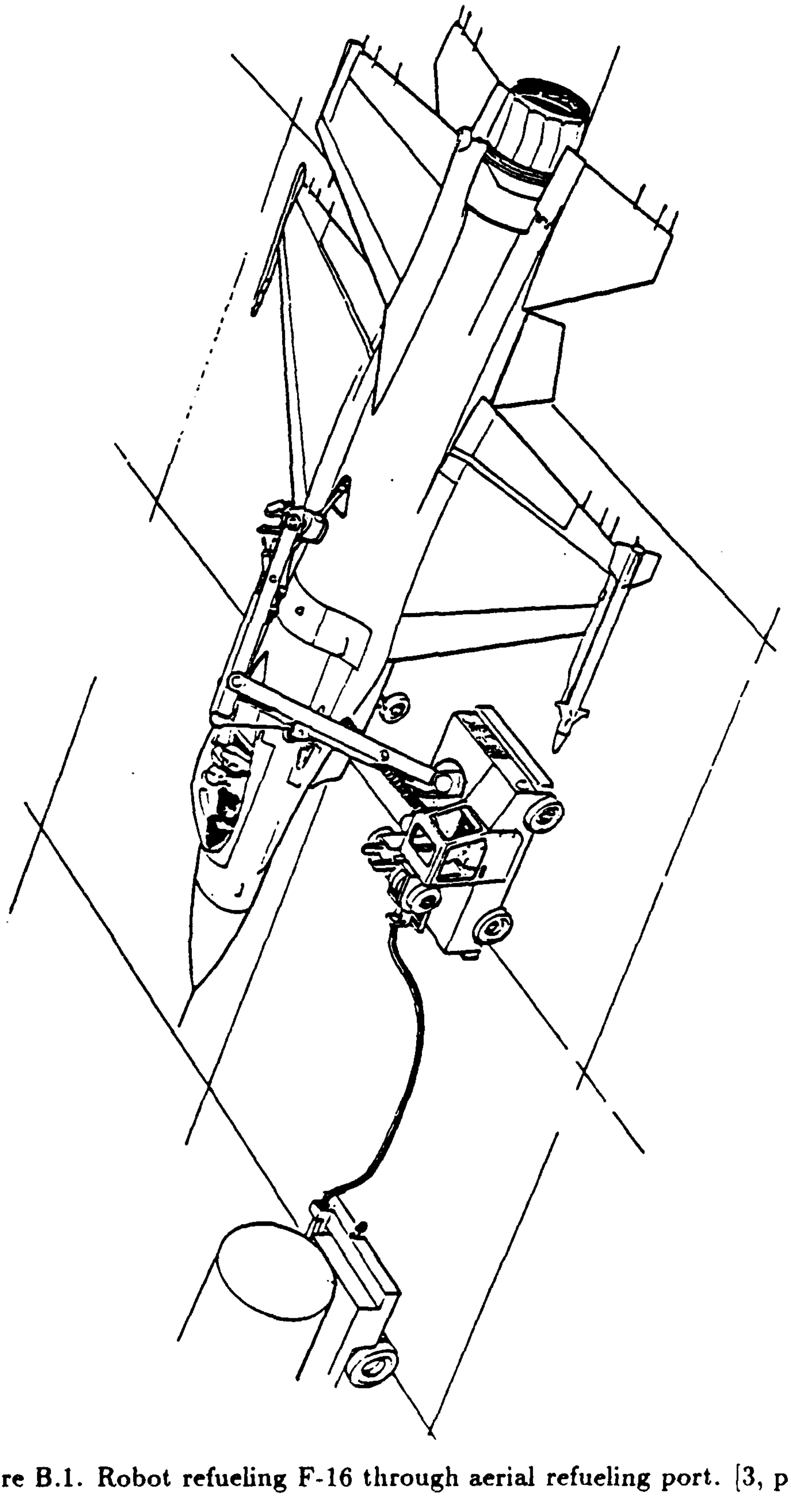


Figure B.1. Robot refueling F-16 through aerial refueling port. [3, p. 27]

3. Induction coils in the port and the nozzle can be used to pass signals between the aircraft systems and the robot.
4. The port is never further back then mid-wing, therefore it can always be reached from the front of the aircraft without requiring the refueling cart to travel underneath the wing or fuselage.
5. The robot does not have to remove any access covers to open the aerial refueling port, although the pilot or ground crew must open the aerial refueling port by lowering the slipway (this could be done just before engine shutdown). In contrast, use of the single point refueling port would require the manipulator to loosen fasteners on the access cover — a fine manipulation task requiring special tooling, and more sophisticated robot capabilities.

Battelle's manipulator concept includes proximity sensors on each link of the arm to prevent inadvertent contact of the arm with the aircraft [3, p. 31]. However, Battelle does not include any provisions for sensing or controlling the contact forces and torques created during nozzle insertion. As a result the Battelle concept could potentially have problems with parts breaking or jamming due to excessive contact force, and the uncontrolled contact forces could also cause instability in the robot arm controller.

The Battelle near-term concept is teleoperated. The operator would assist in positioning the cart, maneuvering the arm to insert the nozzle, monitoring the fuel flow, and checking for leaks. The operator would sit in a protective shelter, either on the cart or in a separate nearby van or mobile shelter. Far-term refueling concepts could be entirely automated, with an operator merely supervising one or more robots from a remote location [3, pp. 14, 37, 41]. A form of compliant motion control would be absolutely necessary for the autonomous far-term concept in order to insure stable application of the correct contact forces. Near-term, teleoperated concepts would also benefit from the use of a compliant motion controller since it would allow the nozzle insertion task to be performed faster, and by less skilled operators.

B.2 Aerial Refueling Port and Nozzle Description.

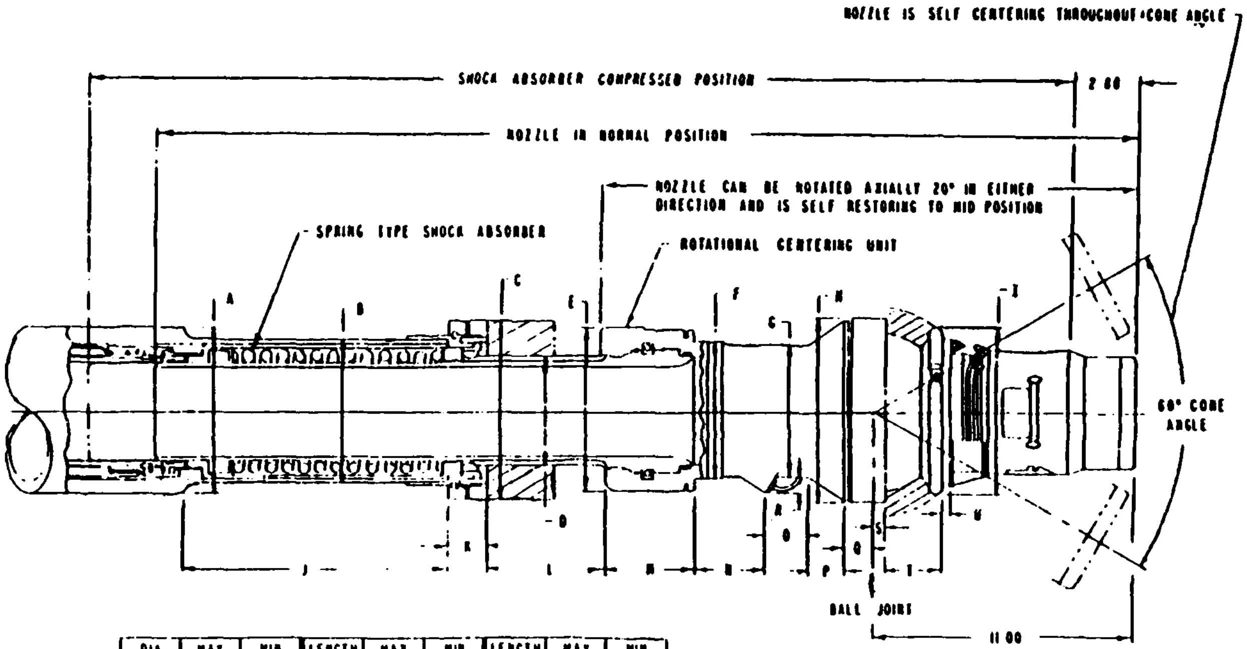
All current U.S. Air Force fixed wing aircraft use a boom type refueling system (helicopters, some U.S. Navy aircraft, and several allied air forces use the probe and drogue system). This system consists of a tanker aircraft mounted boom with a short cylindrical nozzle attached by a ball joint to the end of the boom. The aircraft being refueled must have a receiver port to accept the nozzle. This receiver port may be installed at the end of a trough-like slipway which helps to guide the nozzle into the receiver port.

Because all these different aircraft must interface to the standardized booms and nozzles used by the KC-135 or KC-10 tanker aircraft, their receiver port installations are very similar. Most USAF aircraft designed since the early 1970's have used as a standard the Universal Aerial Refueling Receptacle Slipway Installation (UARRSI) [25], [56]. Those not using the UARRSI may use a modified slipway (e.g. the F-16), or in the case of older aircraft they may not use any slipway at all (e.g. the F-4) [25]. In all cases the critical receiver port dimensions are the same, to within a few hundredths of an inch, so as to remain compatible with the tanker aircraft nozzle [25]. Because of this high degree of standardization, the UARRSI was used as the baseline for this study.

Figure B.2 shows the nozzle attached to the end of the boom, and Figure B.3 provides additional nozzle dimensions. The nozzle is slightly tapered with a maximum diameter of 5.25 inches, and an insertion depth of 6.245 inches [8]. The nozzle is made of high strength stainless steel with a polished finish. Nozzle weight is approximately 45 lbm [25].

The UARRSI slipway opening is approximately 23 inches long by 13 inches wide. The slipway ramp extends downward at an angle of 15 degrees to a point

FED. SUP CLASS
1680



NOZZLE IS SELF CENTERING THROUGHOUT CONE ANGLE
SHOCK ABSORBER COMPRESSED POSITION
2.80
NOZZLE IN NORMAL POSITION
NOZZLE CAN BE ROTATED AXIALLY 20° IN EITHER DIRECTION AND IS SELF RESTORING TO MID POSITION
SPRING TYPE SHOCK ABSORBER
ROTATIONAL CENTERING UNIT
60° CONE ANGLE
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
BALL JOINT
11.00

DIA	MAX	MIN	LENGTH	MAX	MIN	LENGTH	MAX	MIN
A	6.962		J	11.175	10.970	S	.501	.450
B	6.02		K	1.680	1.555	T	2.463	2.386
C	7.43		L	4.959	4.829	U	.31	.30
D	4.63		M	4.010	3.744			
E	6.75*		N	2.96	2.86			
F	5.77		O	1.926	1.673			
G	5.65		P	1.647	1.454			
H	7.76		Q	1.310	1.265			
I	7.05		R	.73	.67			

* DOES NOT INCLUDE SPRING GUARD OR BOLT HEADS.

ALL DIMENSIONS ARE IN INCHES
UNLESS OTHERWISE SPECIFIED, TOLERANCE IS ± .010

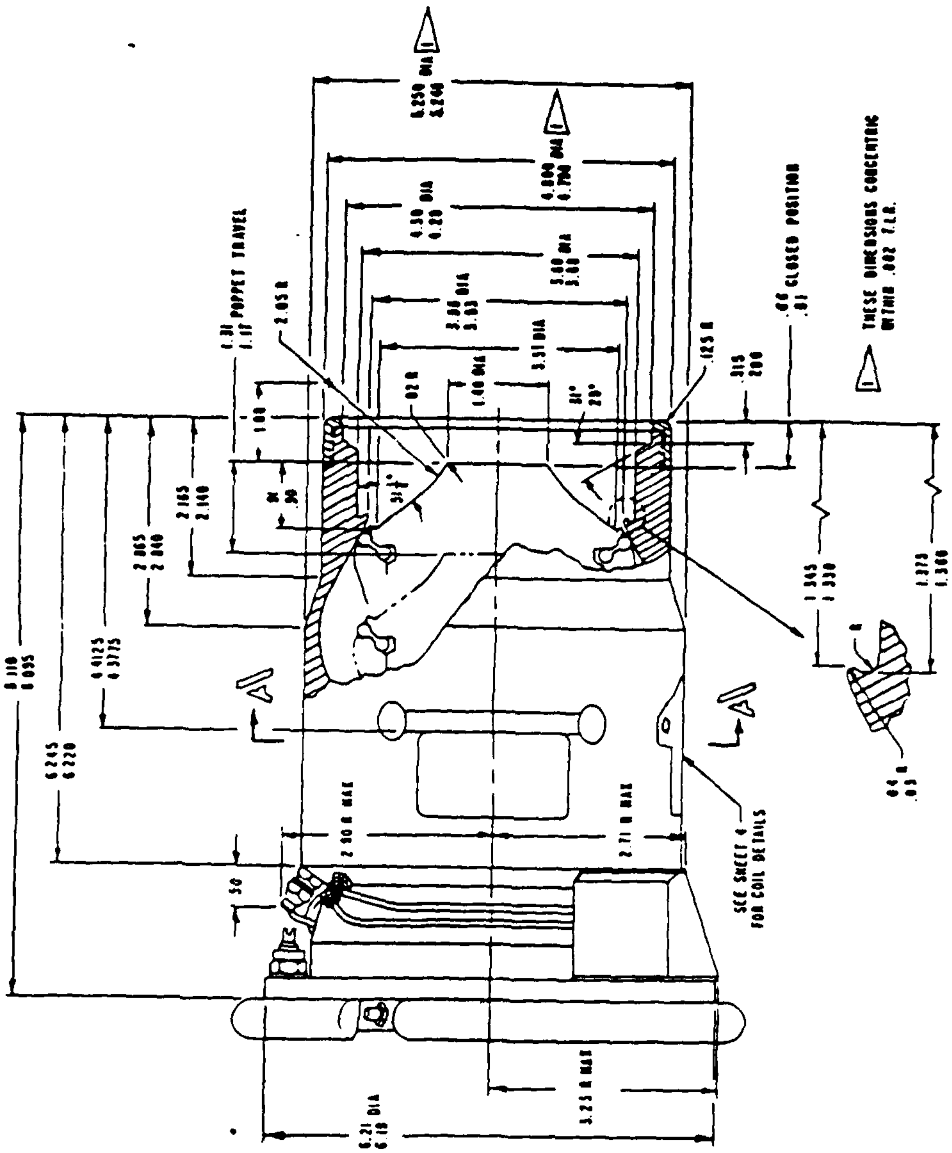
APPROVED 11 AUGUST 69 REVISED

THIS IS A DESIGN STANDARD, NOT TO BE USED AS A PART NUMBER.

P.A. AIR FORCE 11 Other Cust	TITLE NOZZLE-UNIVERSAL AERIAL REFUELING TANKER BOOM	MILITARY STANDARD MS27604(USAF)
PROCUREMENT SPECIFICATION	SUPERSEDES	SHEET 1 OF 4

Figure B.2. Outer End of KC-135 Refueling Boom Including Nozzle [8].

FED. SUP CLASS
1680


6.250 DIA
6.240
0.800 DIA
0.790
4.30 DIA
4.20
1.40 DIA
3.51 DIA
1.31
1.17 POPPET TRAVEL
2.05 R
.02 R
.125 R
.315
.280
.66
.61 CLOSED POSITION
THESE DIMENSIONS CONCENTRIC
WITHIN .002 T.I.R.
1.00
2.165
2.140
2.065
2.040
1.345
1.330
1.375
1.360
8.110
8.095
4.4125
4.3775
6.245
6.220
A
A
2.90 R MAX
2.71 R MAX
SEE SHEET 4
FOR COIL DETAILS
.50
3.25 R MAX
6.21 DIA
6.18

APPROVED 11 AUGUST 69 REVISED

P.A. AIR FORCE 11 Other Cust	TITLE NOZZLE-UNIVERSAL AERIAL REFUELING TANKER BOOM	MILITARY STANDARD MS27604(USAF)
PROCUREMENT SPECIFICATION	SUPERSEDES:	SHEET 2 OF

Figure B.3. Aerial Refueling Nozzle [8].

where it meets a conical lead-in to the receiver port [10]. The receiver port is mounted with the centerline at an angle of 31 to 46 degrees (depending on the type of aircraft) below the surface of the UARRSI. For this study an installation with an angle of 30 degrees was assumed. Figure B.4 indicates the port is slightly tapered, to match the nozzle, with a diameter of 5.305 inches at the opening and a diameter of 4.901 inches at the back

The UARRSI is designed for an optimum nozzle insertion when the approach elevation angle matches the installation angle of the receiver port (in this case 30 degrees). Because of the compliance of the nozzle ball joint, an approach elevation angle error of ±4 degrees can be tolerated. Two types of insertions are possible. A precision insertion may be made, in which case the nozzle is inserted directly into the receiver port with little or no contact with the slipway. Alternately, the nozzle may first contact the slipway, with normal forces of up to 900 lbf, and then slide along the slipway until it is inserted in the port. During this type of insertion, frictional forces are minimized since the contact surfaces of the slipway and the nozzle are unpainted, polished stainless steel [9, pp. 32–33]. As envisioned in this thesis, the proposed robotic system will use the latter insertion technique since it is easier for the vision system to guide the end-effector to contact with the slipway then to place the end-effector precisely in the receiver port.

Once inserted in the receiver port, the nozzle must slide far enough to open poppet valves in both the port and the nozzle. These valves are spring loaded and typically require a force of 60 to 100 lbf to open (maximum specified force is 160 lbf). As the valves are opened, two hooks are hydraulically extended from the sides of the receiver port and engage latches machined into the nozzle [9, p. 37]. Therefore, for a successful refueling the robot's control algorithm must direct the manipulator to exert the same force along the nozzle's longitudinal axis once it is inserted in the receiver port.

Normally, when the nozzle and port are disengaged the latch hooks are pulled

CONFIGURATION	PART NUMBER	A	B	C	D	E FRONT	E REAR	F	G	H	I	J	K	W	L	L₁	M	N	O
A-10 PARKER AARSI	2730526H3	3.500	2.000	2.966 +/-.001	3.191 +/-.001	1.125 +.000 -.001	1.175 +.000 -.001	4.901 +/-.001	5.305 +/-.005	4.807	5.257	.047	.024	.662					
VARIOUS PARKER UARRSI	2670117	3.965	2.000	2.906 +/-.002	3.131 +/-.002	1.000 +.000 -.001	1.000 +.000 -.001	4.901 +/-.001	5.305 +/-.005	4.812	5.262	.045	.022	.635	4.67			5.05	4.54
F-15	2710902	3.800	2.000	2.966 +/-.001	3.191 +/-.001	1.125 +.000 -.001	1.175 +.000 -.001	4.901 +/-.001	5.305 +/-.005	4.807	5.257	.047	.024	.662					
C5A&B	4-1265-101	3.900	1.900	2.966 +/-.001	3.191 +/-.001	1.105 +.000 -.001	1.1256 +.000 -.001	4.901 +/-.001	5.305 +/-.005	4.806	5.277	.043	.024	.615					
F4	VM65350	3.500	1.995	2.907 +/-.0025	3.132 +/-.0025	1.000 +/-.001	1.000 +/-.001	4.960 +.002 -.000	[illegible] +.010 .000	4.814	5.264	.077	.018	[illegible]					
B-52	5-80674	4.06	1.880	3.010 +.0025 -.0015	3.150 +/-.002 3.250 +/-.002	1.035 +/-.001 TOP 1.235 +/-.001 BOT	1.210 +/-.001	4.959 +.010 .003	[illegible] +.010 .000	[illegible]	5.265	.075	.018	.75					
F-111	2-1163-251	3.900	1.910	2.966 +/-.0005	3.191 +/-.0005	1.125 +.000 -.001	1.125 +.000 -.001	4.901 +/-.001	5.302 +/-.005	4.807	5.257	.047	.024	.662					
A-7D	1-1267-451	3.500	2.000	2.966 +/-.001	3.191 +/-.001	1.125 +.000 -.001	1.125 +.000 -.001	4.901 +/-.001	5.305 +/-.005	4.307	5.257	.047	.024	.662					
VARIOUS IAR UARRSI	7034	3.965	2.000	2.906 +/-.002	3.131 +/-.001	1.000 +.000 -.001	1.000 +.000 -.001	4.901 +/-.001	5.305 +/-.005	4.812	5.262	.045	.022	.635	4.65	4.675	.435 ±.001		

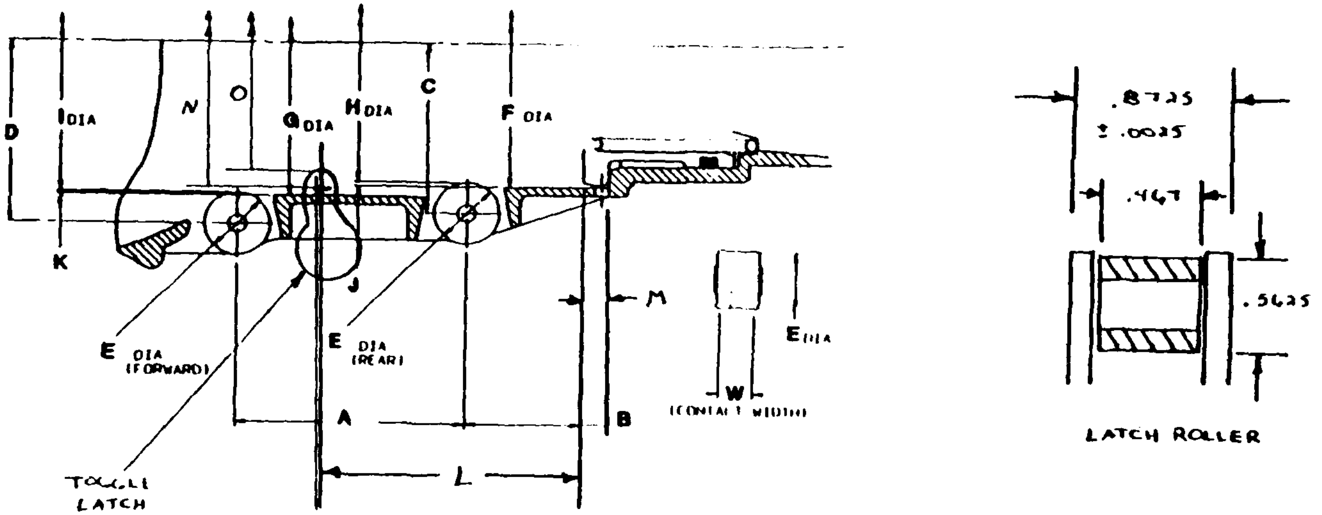


Figure B.4. Aerial Refueling Receptacle. [25]

back into the port and the boom is retracted at rates of up to 10 ft/s. However the nozzle and port can also be disengaged by force if the hooks fail to retract, or the receiver aircraft moves out of the refueling position envelop. The tensile force developed during this second type of disconnect should not exceed 500 lbf [9, p. A-18]. The robot refueling system will probably be required to strictly use the normal disconnect technique, because of the large force required for the alternate technique and the "whip-lash" effect created when the manipulator suddenly broke free.

For a minimum distance of 12 inches around the UARRSI, the aircraft fuselage is hardened to resist inadvertent boom strike loads of up to 1800 lbf normal to the surface. Also, aircraft are typically designed to be free of protrusions (e.g. antennas, canopies, control surfaces) within an area or ten feet by five feet in front of the UARRSI and two feet to either side or aft of the UARRSI [9, p. A-26]. These provisions are fortuitous for the robot refueler since they limit the number of obstacles the robot must avoid, and insure the robot will not damage the aircraft should the end-effector miss the slipway by several inches.

Two other UARRSI features are of particular use to the refueling robot. While the nozzle and port are engaged, induction coils in both the nozzle and port pass electrical signals allowing the receiver aircraft to communicate with the tanker [9, pp. 50–51]. The robot refueler can use this same system to receive signals from fuel tank sensors indicating when the tanks are full. Also, the slipway surface is illuminated by recessed lights to aid in night refueling [9, pp. 27, 42]. The lights can be used by the robot's vision system to help it locate and track the slipway.

Appendix C. *New and Modified Software.*

This appendix contains abstract listings for all new or modified software modules developed for the compliant motion control environment and the implementation of the impedance control law. The actual software listings are compiled in AFIT Robotic Systems Laboratory Report No. 3 [14]. The appendix does not include existing RHCS/R3AGE or ARCADE software which was used in the compliant motion control environment without change (this software is documented in [34] and [33]).

This appendix is organized into three sections. Section C.1 contains routines used at the organizer level. Sections C.2 and C.3 contain routines used at the coordinator level on the Servo and Parallel processors. No new or modified software was required at the hardware level.

C.1 Organizer Software.

This section contains new or modified software used at the organizer level to implement the compliant motion control environment and an impedance controller. The code is written in DEC VAX Fortran Version 4.0 and is targeted to run on a DEC MicroVax [11].

The main module running at the organizer level is FORCER3AGE. Organizer subroutines are listed in Table 4.3.

```
****************************************************************

FORCER3AGE:  MicroVAX II version of the Parallel RAL Real-time
             Robotic Algorithm Exerciser  FORCE CONTROL VERSION:

Abstract:  This program is a derivative of Capt. M. B. Leahy's
  MVIIR3AGE program (Version 1.0, 12 May 88).  The program allows
  use of the R3AGE environment for implementing an impedance based
  compliant motion controller.

  This version runs under MicroVMS and supports the complete
  parallel processor environment although currently only the servo
  and dynamics processors are accessed. Dynamics and servo sampling
  speed, initial conditions, trajectories and load configuration
  are all user selectable. Communication with the parallel
  processors is transparent to the user. The R3AGE user's quide
  provides the user with the detailed information necessary to
  successfully utilize the R3AGE test environment.  The compliant
  motion control features are described in D. Duvall's thesis
  "Compliant Motion Control for Robotic Refueling", December 1988.

VERSION 1.0       DAVID J. DUVALL 6 September 1988

VERSION 2.0       DAVID J. DUVALL        29 September 1988
  Modified to run with MicroVax III as the parallel processor
  instead of a PDP 11/73.  Organizer is now no longer required
  to set-up and initialize the parallel processor -- the user
  must do this independantly.
```

```
****************************************************************

SUBROUTINE FORREOUT(ND,DELT,ETITLE,P$CHAN)

Abstract:  This subroutine allows data to be recovered and
   stored by R3AGE by reading  data from the coordinator.
   All data is formated and stored in ETITLE.dat.

   This routine is uniquely adapted to work with the IMPSER2.MAC
   routine on the PDP; specifically the last data values of
   each data block (ex. Force, Joint Angles, etc.) are ignored
   because IMPSER2 saves one less value (per block) than is
   sent to the MicroVax.

   This subroutine is an adaptation of Capt M. B. Leahy's
   REOUT subroutine (Version 4.1) which was used to recover
   joint position and velocity error data.

VERSION 1.0 DAVID J. DUVALL 15 SEP 88

Inputs:

   ND:    An integer that represents the number of data points.

   DELT:  A real variable which equals the sampling rate.

   ETITLE:  A character string that represents the name of
            the output file; ETITLE.DAT

   P$CHAN:  Integer value used to store channel number.
```

****

SUBROUTINE SLCTTJ(NIC,PNIC,NSPI,ND)

Abstract: This subroutine allows the user to select the manipulator joint space position, velocity and acceleration trajectories for control algorithm evaluation under R3AGE. A zero, slow and fast set of base trajectories are predefined. The user may specify his/her own base trajectories contained in a set of three files. Actual trajectories stored in COMMON arrays are determined from the base trajectory and input sample rate. Position trajectories are formed by addition of the initial conditions selected by the SLCTIC subroutine and actual trajectory data, and are checked against specific manipulator range limits by the RCHK subroutine. Trajectories starting from IC option 2 are reversed. The option to leave existing actual trajectory data unaltered is also available.

VERSION 2.1 DAVID J.DUVALL 2 NOV 88

Commented out all options but the choice of the user defined trajectory. Subroutine is now uniquely adapted to the cartesian control task. Disabled trajectory reversing for initial condition #2 if the user defined trajectory is selected.

VERSION 2.0 DAVID J. DUVALL 15 SEP 88

For the "User Defined" trajectory option (Option 3):

(1) Acceleration trajectories are not read.

(2) Position trajectories are not checked to see if they

violate the PUMA's workspace limits.

(3) Position trajectories are intended to be used as read from the data file. They are not added to the initial condition values.

This version of SLCTTJ is intended for reading cartesian position and velocity trajectories required for impedance control.

VERSION 1.0 MICHAEL B. LEAHY JR. 7 DEC 85

REVISION 1: 30 JAN 86	Incorporates the changes necessary so that IC2 initial condition selection is correctly handled when an unchanged trajectory is selected.
REVISION 2: 26 FEB 86	Incorporates TMODE into MTYPE common.
REVISION 3: 27 MAR 86	Incorporates changes to allows generation of zero trajectory for any 7ms multiple.
REVISION 4: 8 Aug 86	Corrects errors in trajectory file specification read statements.
REVISION 5: 22 FEB 88	Change default fast trajectory to spline one.

Input:

QOR: A (6x1) COMMON vector of initial joint angles in radians.

NSPI: An integer representing sampling rate speed.

NIC: An integer representing initial condition number.

PNIC: An integer representing the previous initial condition number.

Output:

ND: An integer representing the number of sampling points.

QDSI: A (6,ND) COMMON matrix of incremental joint positions.

QDST: A (6,ND) COMMON matrix of joint velocities.

QDSTT: A (6,ND) COMMON matrix of joint accelerations.

****

SUBROUTINE SLCTIC(NIC)

Abstract: This subroutine allows the user to select the manipulator initial condition for control algorithm evaluation under R3AGE. The IC may be one of three predefined conditions, input by the user or remain unchanged. User input conditions are automatically checked against the specific manipulator range limits by the RCHK subroutine. IC values are stored in COMMON vectors in degrees and radians.

VERSION 2.0 DAVID J. DUVALL 1 OCT 88

Redefined initial condition choices to adapt to FORCER3AGE type tasks.

VERSION 1.0 MICHAEL B. LEAHY JR. 7 DEC 85

REVISION 1: Incorporates TMODE into MTYPE common and corrects
27 FEB 86 error of missing T6D matrix in TRAJ common.

Output:

QO: A (6x1) COMMON vector of initial joint angles in degrees.

QOR: A (6x1) COMMON vector of initial joint angles in radians.

NIC: An integer representing IC option number. When the IC's remain unchanged so does this value.

```
******************************************************************

 SUBROUTINE CALIBCON(FCONST)

 Abstract:  Organizer level subroutine to read in and calculate the
    the constants needed by the Macro subroutines CFINT, CFORCE,
    KIN62, and for initial calibration of the PUMA.  Results of this
    subroutine must be passed to the PDP 11/73 before they can
    be used.

 VERSION 1.0 DAVID J. DUVALL 1 SEP 88

 INPUTS:  None from calling routine. This subroutine calls the file
    CFINTINP.DAT to get tool data, sensor scaling information,
    and force & moment limits.

 OUTPUTS:

    FCONST -- A 12x1 real vector containing the constants.
```

****

SUBROUTINE ADDCON(CONST)

Abstract: Organizer level subroutine to read in and calculate the the constants needed by the Macro subroutine MAXFOR, and by the servo processor for calculating the control torques. Results of this subroutine must be passed to the PDP 11/73 before they can be used.

VERSION 1.0 DAVID J. DUVALL 1 SEP 88

INPUTS: None from calling routine. This subroutine calls the file CFINTINP.DAT to get force & moment limits, and threshold values for the force & moment deadband function. The routine also calls the file GAININP.DAT to get Kp & Kv gain values and desired angular positions for joints 1,4,5, and 6.

OUTPUTS:

CONST -- A 23x1 real vector containing the constants.

C.2 Coordinator Software for the Servo Processor.

This section contains software used to implement the compliant motion control environment at the coordinator level on the Servo processor. The software is written in DEC PDP assembly code for the RSX11 compiler. The code is targeted to run on a PDP 11/73.

The program IMPSER2 is the main module for the Servo processor. Other Servo processor subroutines are listed in Table 4.4.

```
**************************************************************

IMPSER2: Impedance Control Macro, Servo Processor, 2 Degrees of Freedom

 Abstract: This program controls a robotic manipulator by combining
      the elements of Hogan's Impedance Control law to determine the
      control torque for links 2 and 3 of the manipulator.  The other
      4 links are controlled by a PD gain loop.  Impedance coefficents
      are provided by a parrallel processor at user defined intervals.

     The user interfaces to this program through the PFMAGE subroutine
     which also sets up the arm for testing and calibrates the force
     sensor.

     Position and velocity errors are output to the organizer by
     FHDVAX.

   VERSION 1        DAVID J. DUVALL 31 Aug 88

   VERSION 2.0      DAVID J. DUVALL                 28 Sep 88
     Corrected errors in assumptions about diagonal nature of
     impedance coefficent matrices.  Can now handle fully
     populated impedance coefficent matrices (IJM, IJMBJ).
     Also modified to work with MicroVax as a parrallel processor
     (using VaxLab) instead of another PDP 11/73.

   VERSION 3.0      DAVID J. DUVALL 3 Nov 88
     Set force sensor sample interval to be .014 seconds or the Servo
     processor sample interval, whichever is longer.  This enables
     Servo processor sampling at intervals under .014 seconds.  Also,
```

when the Servo processor sample time is less than .014 seconds, test data (force, position, etc) is only saved on every other pass through the servo control loop; this avoids accumulating so much data it overwrites other memory areas.

NOTE: Version 3.0 has not been successfully run at times of less than .014s because of the computation time required in the main loop (slightly over .007s).

****

FHDVAX: Force (and joint angle) history data to VAX.

Abstract: This macro transmits interface force, and angular position data from a control algorithm to the VAX.

VERSION 1.0 DAVID J. DUVALL 14 SEP 88

CALLING FORMAT: CALL FHDVAX(NPNTS)

Input:

NPNTS: The number of data points in the buffer.

****

GFORCE: Get force and moment data from the JR3 sensor through the DRV11 card.

Abstract: This routine gets a 11 byte packet of force/moment data the JR3 force sensor through a DRV11 card. Data is requested by asserting CSR0. Data Valid is detected by REQ B being asserted. After data has been received CSR0 is cleared and the routine waits until REQ B falls before initiating another data collection sequence. The address of the force/moment data buffer is passed into this routine so that the user can choose where the data will be stored. The error flag is set if the JR3 does not respond in .9 millisec.

VERSION 1.0 Michael B. Leahy Jr. 12 July 88

VERSION 1.1   DAVID J. DUVALL 29 OCT 88

ROUTINE NOW CLEARS THE DRV11 ADDRESS AFTER READING THE NULL WORD. THIS CLEARS ANY "GLITCHES" WHICH MIGHT BE PRESENT ON THE PDP 11/73 END OF THE SERIAL LINK. TIMEOUT TIME IS NOW SET TO .24 MILLISEC.

CALLING FORMAT: CALL GFORCE(DATA,ECODE)

Input:

DATA: An integer variable which represents the address of the data to be stored.

Output:

ECODE:	An integer variable which is set if a time out error occurs in the DRV11 communication

** ********

SUBROUTINE CFORCE: Convert JR3 DMA Data to Measured Force and Moment

ABSTRACT: This subroutine converts the force and moment data supplied by the JR3 force sensor over the DMA interface to measured forces and moments. The JR3 supplies a signed number of counts (+-2048 max) indicating external force on the sensor and including constant bias errors. The counts must be scaled by a user defined scaling factor

Scale Factor = (Max Allowed Force)/2048

and then the calibration constant is used to eliminate any constant bias in the force data

Measured Force = (Force Data)*(Scale Factor) - Calibration Force

The measured forces and moments resulting from this routine are in the force sensor frame.

This subroutine requires the JR3 force and moment data to be stored in 6 consecutive buffers (12 words) in the calling program. The scale factors are computed at the organizer level before run time and must also be stored in 6 consecutive buffers. Lastly, the calibration forces and moments must be arranged in 6 consecutive buffers.

VERSION 1.0 DAVID J. DUVALL 21 AUG 88

CALLING FORMAT: CFORCE(DATA,SCALE,CALIB,FMEAS)

INPUT:

DATA -- Starting address of the 6 buffers containing the JR3 force and moment data (this is the 3rd word of the 11 words sent by the sensor).

SCALE -- Starting address of the 6 buffers containing the user defined scale constants.

CALIB -- Starting address of the 6 buffers containing the calibration forces and moments.

FMEAS -- Starting address of the 6 buffers where output (measured forces and moments) should be stored.

OUTPUT: The FMEAS addressed buffer, and the next 5 buffers after FMEAS contain the measured forces and moments in the sensor frame.

****

SUBROUTINE MAXFOR: Macro Subroutine to Check for Excess Force on Sensor

ABSTRACT: This subroutine checks to insure the measured forces on the JR3 force sensor do not exceed the user defined force and moment limits. The user defines the limits in terms of percentages of the maximum allowed force or moment specified by JR3. These percentages are converted by the organizer into the force and moment limits used here.

VERSION 1.0 DAVID J. DUVALL 22 AUG 88

CALLING FORMAT: MAXFOR(FMEAS,LIMITS,ECODE)

INPUT:

FMEAS -- Starting address of the 6 buffers where measured forces and moments are stored.

LIMITS -- Starting address of the 6 buffers where the force and and moment limits are stored.

ECODE -- Address of the buffer where any error code shoulb be placed

OUTPUT: The ECODE addressed buffer contains the value 11 (octal) if any of the force or moment limits are exceeded.

****

CFINT: Calculate Interface Force

Abstract: This assembly language subroutine computes the interface force* (Fint) in world coordinates from the measurements of external force on the tool in sensor coordinates. Moments are taken about the end of the tool in the nsap frame. The nsap frame is assumed parrallel to the force sensor frame but displaced from the force sensor frame by a distance LTS along the +z axis of the force sensor. The subroutine also eliminates gravitational force from the force sensor measurements. Before using this subroutine the force sensor data must be scaled and calibrated (see CFORCE).

Interface force is used in Hogan's 1985 impedance control law and is defined as being equal and opposite to the external force applied by the environment to the tool (gravity is not a force applied by an object in the environment).

*Note: Force includes both forces and moments

VERSION 1.0   DAVID J. DUVALL   12 AUG 88

VERSION 1.1   DAVID J. DUVALL   29 OCT 88

Switched signs on FCON2 to conform with sign convention for LCGS used in thesis writeup. LCGS is now defined positive in the negative a-vector direction (negative joint 6 z-axis).

DOCUMENTATION: See D. Duvall's Thesis (I. 79), "Compliant Motion Control for Robotic Aircraft Refueling"

CALLING FORMAT: CALL CFINT(FIX)

Inputs:

FIX An integer address of the buffer where the interface force should be stored.

Outputs: The FIX addressed buffer contains the interface force and moment values stored in the following order:

FIX, FIY, FIZ, MIX, MIY, MIZ

NOTE: The first four Floating Point Accumulators are used. This routine requires the 18 elements of the 6 DOF kinematics coordinate rotation matrix (nsa). Constants FCON1, FCON2 and LTS are assumed to be globally declared. Measured forces are assumed to be stored in global variables: FMX, FMY, FMZ, MMX, MMY, MMZ.

****

KIN62: Six DOF Kinematics for PUMA, with Px and Pz Position Vectors

Abstract: This assembly language subroutine computes the rotation matrix for transforming coordinates from the 6th link (tool) frame to the world frame (nsa matrix) for the PUMA manipulator with 6 degrees of freedom (DOF). The x and z elements of the position vector are also determined, but are based on only 2 DOF (Q1=Q4=Q5=Q6=0, i.e. only links 2 and 3 are free to rotate). The routine uses the equations presented in Fu, Gonzalez, and Lee (p.45). The routine also calculates the trignometric function values for the 6 joint angles.

This routine can be used to set up the rotation matrix for transforming forces from the sensor frame to the world frame assuming the sensor frame is aligned parrallel to the 6th link frame. The routine also will provide cartesian x & z position for 2 DOF PUMA manipulator.

VERSION 1.0 DAVID J. DUVALL 28 JUL 88

Original version, used only for calculating nsa matrix without determining cartesian position. Formerly called KIN6.MAC.

VERSION 2.0 DAVID J. DUVALL 30 AUG 88

Incorporated provisions for calculating the Px and Pz elements of the cartesian position vector assuming only links 2 and 3 are free to rotate and all other links are locked at Q=0.

DOCUMENTATION: See D. Duvall's Thesis (I. 79), "Compliant Motion Control for Robotic Aircraft Refueling"

CALLING FORMAT: CALL KIN62(Q,POS)

Inputs:

Q	The address of the buffer containing the 6 joint angles in radians.
POS	The first address of the 2 buffers where the values of Px and Pz should be placed.

Outputs: The 18 elements of the nsa matrix are calculated and stored in the global variables:

NX, NY, NZ, SX, SY, SZ, AX, AY, AZ
(3 column vectors -- n, s, a)

The 2 elements of the position vector are stored at POS in the order: Px, Pz.

NOTE: Only the first 4 floating point accumulators are used. The calling routine must supply a cosine table at the global variable location COSV. The values of the constants A2, A3, and D64 must also be globally declared.

****

PFMAGE: Macro algorithm exerciser: parallel force version

Abstract: This subroutine interfaces the users coordinator level control algorithm to the organizer level RAGE program. PFMAGE sets up the arm for testing by doing the following: receive and store the COS table, receive and store the test position, and velocity trajectories. Move the arm to the ready position and calibrate the force sensor, placing the calibration values in a global buffer. Move the arm to the test position, send that position to the organizer and then receive the position trajectory data. Once the organizer sends the GO command control is returned to the calling algorithm.

This subroutine is a modified version of Capt M. B. Leahy's PMAGE macro, Version 4.1.

VERSION 1.0   DAVID J. DUVALL 2 SEP 88

CALLING FORMAT: CALL PFMAGE(QANG,CTABLE)

Inputs:

QANG: An integer variable representing the starting address of a (6x1) real vector of joint angles.

CTABLE: A variable representing the starting address of a 361 position table of real cosine values.

NOTE: All routines that call this subroutine are stopped by this routine.

C.3 Coordinator Software for the Parallel Processor.

This section contains software used to implement the compliant motion control environment at the coordinator level on the Parallel processor. The software is written in VMS Fortran and uses VAXLab communications functions [11],[12],[13]. It is targeted to run on a DEC MicroVax III.

The program PCIMPVAX is the main module for the Parallel processor. Other Parallel processor subroutines are listed in Table 4.5.

```
****************************************************************

PCIMPVAX: Calculate the impedance coefficents for a robotic
          manipulator.

Abstract:  This program performs the impedance coefficent
  calculations for the FORCER3AGE environment.  VaxLab utilities
  are used to allow a MicroVax III (RVS2A) to function as a
  parrallel processor and calculate the impedance coefficents
  in real time.  The DRV11-J is initialized and employed to
  communicate between RVS2a and the servo processor. Constants
  required for the impedance coefficent calculations are calculated
  prior to the real-time loop.

VERSION 1.0          DAVID J. DUVALL  29 SEP 88
```

****

SUBROUTINE IMPCONST(CONST,MINV,BMAT)

Abstract: This subroutine calculates the 25 constants used by the Fortran subroutine CIMPG2VAX in determining the impedance coefficent matrices and the gravity vector for links 2 and 3 of the PUMA arm. Payload is assumed to be a point mass with center of gravity along the joint 6 axis. The desired mass and the damping (BMAT) matrices are assumed to be diagonal. Joints angles 4,5, and 6 are assumed to be set to zero.

VERSION 1.0 DAVID J. DUVALL 4 AUGUST 1988

INPUTS: None.

OUTPUTS:

CONST -- A 17x1 real vector containing the constants.

MINV -- A 2x2 matrix containing the elements of the inverse desired mass matrix (diagonal).

BMAT -- A 2x2 matrix of damping coefficents (diagonal).

Note: This subroutine calls the file CIMPG2INP2.DAT to get PUMA

Denavit-Hartenburg and inertial parameters, and payload inertial parameters.

****

SUBROUTINE CIMPG2VAX(Q,CONST,MINV,BMAT,IMPCOF)

ABSTRACT: This subroutine calculates the impedance coefficents and gravity for 2 degrees of freedom for the PUMA 560 robot arm. Joints 4,5, and 6 are assumed fixed at Q=0.0 degrees. Impedance coefficents are based on Hogan's impedance control law. They are:

1. The IJ^-1M^-1 (IJM) Term. The product of the inertia tensor, inverse jacobian, and inverse desired mass matrix (assumed diagonal).

2. The IJ^-1M^-1BJ (IJMBJ) Term. The product of the IJM term and B, the diagonal matrix of damping coefficents, and J, the Jacobian.

3. The J^T+IJ^-1M^-1 (JTIJM) Term. The sum of the Jacobian Transpose (J^T or JT), and the IJM term. The result is a fully populated 2x2 matrix.

Included with the impedance coefficents is the 2x1 gravity vector (GG).

The inertia and gravity equations were developed using Tarn's dynamics and L. Tellman's MACSYMA code. The inverse Jacobian was developed on MACSYMA using Leu and Hemati's dynamic equations generator.

VERSION 1.0 DAVID J. DUVALL 28 SEPT 88

VERSION 2.0 DAVID J. DUVALL 29 OCT 88

Changed the JTIJM term from being the difference of the Jacobian transpose and the IJM term, to being the sum of the Jacobian transpose and the IJM term. This is different from Hogan's implementation but supported by the mathematical development in my thesis. The change should result in negative force feedback (stable) instead of positive force feedback (unstable).

INPUTS:

Q -- The 2x1 vector of joint angles for links 2 and 3.

CONST -- The 17x1 vector of constants needed to compute the impedance coefficents and gravity.

MINV -- The 2x2 diagonal inverse desired mass matrix.

BMAT -- The 2x2 diagonal matrix of damping coefficents.

OUTPUTS:

IMPCOF -- The 14x1 vector of impedance coefficents. IMPCOF is arranged in the following order:

IJM[1,1],IJM[1,2],IJM[2,1],IJM[2,2],IJMBJ[1,1],
IJMBJ[1,2],IJMBJ[2,1],IJMBJ[2,2],JTIJM[1,1],
JTIJM[1,2],JTIJM[2,1],JTIJM[2,2],GG[1],GG[2]

Bibliography

1. An, Chae H. and John M. Hollerbach. "Dynamic Stability Issues in Force Control of Manipulators," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation.* Vol 2, pp. 890–896, 1987.
2. An, Chae H. and John M. Hollerbach. "Kinematic Stability Issues in Force Control of Manipulators," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation.* Vol 2, pp. 897–903, 1987.
3. Battelle Corp.. *Concept Development for Robotic Aircraft Turnaround: Draft Task II Report,* Columbus OH: Battelle Columbus Laboratories, 26 May 1987.
4. Benton, Ronald and Donald Waters. *Intelligent Task Automation, Phase I: Final Report,* for Period October 1982 - December 1985. AFWAL-TR-86-4122, Contract F33615-82-C-5092. Minneapolis, MN: Honywell Inc. Technology Strategy Center, December 1986 (AD-B111560).
5. Chirlian, Paul M. *Basic Network Theory.* New York: McGraw-Hill Book Company, 1969.
6. Craig, John J. *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley Publishing Company, 1986.
7. D'Azzo, John J., and Constantine H. Houpis. *Linear Control System Analysis and Design, Conventional and Modern.* New York: McGraw-Hill Book Company, 1981.
8. Dept. of the Air Force. *Military Specification: Nozzle, Universal Aerial Refueling, Tanker Boom.* MS-27604(USAF). Washington: HQ USAF, 11 August 1969.
9. Dept. of the Air Force. *Military Specification: Aerial Refueling Receiver Subsystems.* MIL-A-87166(USAF). Washington: HQ USAF, 2 May 1983.
10. Dept. of the Air Force. *Drawing: Universal Aerial Refueling Receiver Installation (UARRSI).* USAF/XAR-7034. Washington: HQ USAF, dated ≈1978.
11. Digital Equipment Corp.. *Programming in VAX FORTRAN, Version 4.0.* Maynard MA: Digital Equipment Corp., September 1984.
12. Digital Equipment Corp.. *VAX Realtime User's Guide.* Maynard MA: Digital Equipment Corp., November 1986.
13. Digital Equipment Corp.. *VAXlab Version 1.2.* Maynard MA: Digital Equipment Corp., December 1987.

14. Duvall, David J. *Source Code Listing for Initial Implementation of a Compliant Motion Control Environment and Impedance Controller.* Internal Report AFRSL No. 3. AFIT Robotic Systems Laboratory, Dept. of Electrical Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, December 1988.

15. Drake, S. H. "Using Compliance in Lieu of Sensory Feedback for Automatic Assembly," PhD. dissertation, Department of Mechanical Engineering, Massachusetts Institute of Technology; Cambridge, MA: September 1977.

16. Fu, King-Sun, Rafael C. Gonzalez, and C.S. George Lee. *Robotics: Control, Sensing, Vision, and Intelligence.* New York: McGraw-Hill Book Company, 1987.

17. Hogan, Neville. "Impedance Control: An Approach to Manipulation, Part I - Theory," *ASME Journal of Dynamic Systems, Measurement, and Control,* Vol 107, pp. 1-7, (March 1985).

18. Hogan, Neville. "Impedance Control: An Approach to Manipulation, Part II - Implementation," *ASME Journal of Dynamic Systems, Measurement, and Control,* Vol 107, pp. 8–16, (March 1985).

19. Hogan, Neville. "Impedance Control: An Approach to Manipulation, Part III - Applications," *ASME Journal of Dynamic Systems, Measurement, and Control,* Vol 107, pp. 17–24, (March 1985).

20. Hogan, Neville. "Stable Execution of Contact Tasks Using Impedance Control," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation.* Vol 2, pp. 1047-1054, 1987.

21. Hollerbach, John M. "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. SMC-10, No. 11, pp. 730–736 (November 1980).

22. Houpis, Constantine H. and Gary B. Lamont. *Digital Control Systems.* New York: McGraw-Hill Book Company, 1985.

23. Integrated Systems Inc.. *MATRIXX User's Guide, Engineering Analysis and Control Design, Version 6.0.* Santa Clara CA: Integrated Systems Inc., May 1986.

24. JR3 Inc. *Universal Force-Moment Sensor System Operation Manual.* JR3 Inc, Woodland, CA, September 1985.

25. Kalt, Dexter, H., Technical Specialist, Fuel & Hazards Branch, Flight Systems Directorate (ENFEF). Personal Interview. Aeronautical Systems Div., Wright-Patterson AFB OH, 4 November 1988.

26. Kazerooni, H. and others. "Robust Compliant Motion for Manipulators, Part I: The Fundamental Concepts of Compliant Motion Control," *IEEE Journal of Robotics and Automation,* Vol RA-2, No. 2, pp. 83-94 (June 1986).

27. Kazerooni, H. and others. "Robust Compliant Motion for Manipulators, Part II: Design Method," *IEEE Journal of Robotics and Automation,* Vol RA-2, No. 2, pp. 93–105 (June 1986).

28. Kazerooni, H. "Direct Drive Active Compliant End Effector (Active RCC)," *IEEE Journal of Robotics and Automation,* Vol RA-4, No. 3, pp. 324–333 (June 1988).

29. Kazerooni, H. and T. I. Tsay. "Stability Criteria for Robot Compliant Maneuvers," *Proceedings of the 1988 IEEE International Conference on Robotics and Automation.* Vol 2, pp. 1166–1172, 1988.

30. Khatib, Oussama. "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation,* Vol RA-3, No. 1, pp. 43–53 (February 1987).

31. Lawrence, Dale A. "Impedance Control Stability Properties in Common Implementations," *Proceedings of the 1988 IEEE International Conference on Robotics and Automation.* Vol 2, pp. 1185–1190, 1988.

32. Leahy, Michael B. Jr. *Performance Characterization of a PUMA 600 Robot.* Internal Report No. RAL-56. Robotics and Automation Laboratory, Dept of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy NY, May 1985.

33. Leahy, Michael B. Jr. *The RAL Real-Time Robotic Algorithm Exerciser User's Guide, Version 1.0.* Internal Report No. RAL-61. Robotics and Automation Laboratory, Dept of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy NY, November 1985.

34. Leahy, Michael B. Jr. *The RAL Hierarchial Control System User's Guide, Version 1.0.* Internal Report No. RAL-67. Robotics and Automation Laboratory, Dept of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy NY, April 1986.

35. Leahy, M. B. Jr and others. "Efficient Dynamics for a PUMA 600," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation,* Vol 1, pp. 519–24, 1986.

36. Leahy, Michael B. Jr., and G. N. Saridis. *Compensation of Unmodeled PUMA Manipulator Dynamics, Part II.* Internal Report No. RAL-87. Robotics and Automation Laboratory, Dept of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy NY, September 1986.

37. Leahy, Michael B. Jr. "Dynamics Based Control of Vertically Articulated Manipulators," *Proceedings of the 1988 IEEE International Conference on Robotics and Automation,* Vol 2, pp. 1048–1053, 1988.

38. Leahy, Michael B. Jr., and G. N. Saridis. *Compensation of Industrial Manipulator Dynamics, Part I.* Internal Report No. RAL-101. Robotics and Automation Laboratory, Dept of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy NY, September 1987, To be published in the International Journal of Robotics Research.

39. Leahy, Michael B. Jr. *Robotic Manipulator Control Performance Evaluation.* PhD dissertation. Rensselaer Polytechnic Institute, Troy NY, August 1986.

40. Leahy, Michael B. Jr. Class handout distributed in EENG 540, Robotic Fundamentals. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, Fall Quarter 1987.

41. Leu M. C. and N. Hemati. "Automated Symbolic Derivation of Dynamic Equations of Motion for Robotic Manipulators," *ASME Journal of Dynamics, Measurement, and Control,* Vol. 108, pp. 172-179 (September 1986).

42. Mason, Mathew T. "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. SMC-11, No. 6, pp. 418–432, (June 1981).

43. Murphy, Steve. "Control of Manipulator Orientation in Cartesian and Force Control," Unpublished paper. (April 1988).

44. Paul, Richard P. "Problems and Research Issues Associated with the Hybrid Control of Force and Displacement," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation.* Vol 3, pp. 1966-1971, 1987.

45. Pennington, Jack E. "Space Telerobotics: A Few More Hurdles," *1986 IEEE Conference on Robotics and Automation,* Vol. 2, pp. 813–816, 1986.

46. Potkonjak, Veljko and Miomir Vukobratovic. "Dynamics of Manipulation Mechanisms with Constrained Gripper Motion. Part I," *ASME Journal of Robotic Systems,* Vol 3, pp. 321-334 (March 1986).

47. Raibert, M. H. and J. J. Craig. "Hybrid Position/Force Control of Manipulators," *ASME Journal of Dynamic Systems, Measurement and Control,* Vol. 102, pp. 126–133, (June 1981).

48. Ramming John E. *Apparatus for Simultaneous Measurement of Mutually Perpendicular Forces and Moments.* United States Patent. Patent No. 4,488,441. Assigned to JR3 Inc., Woodland CA, 18 December 1984.

49. Salisbury, J. Kenneth. "Active Stiffness Control of a Manipulator in Cartesian Coordinates," *Proceedings of the 19th IEEE Conference on Decision and Control*, Vol 1, pp. 95-100, 1980.

50. Shimano, B. and B. Roth. "On Force Sensing Information and Its Use in Controlling Manipulators," *Proceedings of the Eighth International Symposium on Industrial Robots,* pp. 119-126, 1976.

51. Symbolics Inc.. *VAX UNIX MACSYMA Reference Manual.* Cambridge MA: Symbolics Incorporated, 1985.

52. Tarn, T. J. and A. K. Bejczy. "Dynamic Equations for PUMA 560 Robot Arm," Robotics Laboratory Report SSM-RL-85-02, Dept. of Systems Science and Mathematics, Washington University; St. Louis, MO, July 1985.

53. Tellman, Larry D.. *Multiple Model Based Robot Control: Development and Initial Evaluation.* AFIT/GE/ENG/88D-55, Air Force Institute of Technology, Air University, December 1988.

54. Townsend, William T. and J. Kenneth Salisbury. "The Effect of Coulomb Friction and Stiction on Force Control," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Vol 2, pp. 883–889, 1987.

55. Unimation Incorporated. *Unimate PUMA Mark II Robot, 500 Series Equipment Manual for VAL II and VAL PLUS Operating Systems.* Manual No. 398U1. Unimation Incorporated (Division of Westinghouse), Danbury CT, March 1985.

56. USAF Aeronautical Systems Div., Flight Systems Directorate, Fuel & Hazards Branch (ENFEF). *Technical Exhibit: Universal Aerial Refueling Receptacle Slipway Installation.* Wright-Patterson AFB OH, 1 June 1977 (amended 9 April 1981).

57. Walker, M. W. and D. E. Orin. "Efficent Dynamic Computer Simulation of Robotic Mechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control,* Vol 104, pp. 205–211 (September 1982).

58. Whitney, Daniel E. "Historical Perspective and State of the Art in Robot Force Control," *The International Journal of Robotics Research*, Vol 6, No. 1, pp. 3–13, (Spring 1987).

Vita

Captain David J. Duvall was born on [redacted] and graduated in 1979 [redacted] Shortly thereafter he entered the United States Air Force Academy where he earned a Bachelor of Science degree in Astronautical Engineering. He graduated from the Academy and was commissioned in the USAF on 1 June 1983. His first assignment was to Los Angeles AFS, California, where he served as a launch vehicle engineer in the Air Force Space Division, Launch & Control Systems Deputate. In 1986 he married Lori Ann Schumacher of Moscow, Idaho. He entered the AFIT School of Engineering's Department of Aeronautical and Astronautical Engineering in May 1987.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA202716

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GA/ENG/88D-1

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Engineering

6b. OFFICE SYMBOL *(If applicable)*
AFIT/ENG

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State, and ZIP Code)*
Air Force Institute of Technology
WPAFB, OH 45433-6583

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING / SPONSORING ORGANIZATION AFWAL Flight Dynamics Lab, Special Projects

8b. OFFICE SYMBOL *(If applicable)*
AFWAL/FIEM

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State, and ZIP Code)*
Wright-Patterson AFB, Ohio, 45433

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
ROBOTIC COMPLIANT MOTION CONTROL FOR AIRCRAFT REFUELING APPLICATIONS

12. PERSONAL AUTHOR(S)
David J. Duvall, Captain , USAF

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM ________ TO ______

14. DATE OF REPORT *(Year, Month, Day)*
1988, December

15. PAGE COUNT
219

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	09	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
ROBOT, ROBOTICS, ROBOT CONTROL, COMPLIANT MOTION, FORCE CONTROL, FORCE FEEDBACK, IMPEDANCE CONTROL

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Thesis Chairman: Michael B. Leahy, Captain, USAF
Assistant Professor of Electrical Engineering

Abstract on Reverse Side.

10 Jan 89

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Capt. Michael B. Leahy

22b. TELEPHONE *(Include Area Code)*
(513) 255-6027

22c. OFFICE SYMBOL
AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Abstract

A promising Air Force application of robotics technology is the refueling of aircraft or spacecraft with a robotic manipulator. The refueling task is fundamentally a component assembly task, and like many other assembly tasks it requires compliance between the manipulator and the environment for success. Compliant motion control techniques, such as impedance control, use force feedback to generate active compliance in the robot manipulator. In this thesis a compliant motion control environment was established, and a simplified, preliminary version of an impedance control law was implemented. The compliant motion environment employs three digital processors in a hierarchial control structure to command a PUMA 560 robot arm. Applied force and moment information are provided by a wrist mounted, three axis force sensor. An original method was developed to transform forces and moments acting on the tool, and measured in the sensor frame, to the cartesian world coordinate frame. This method eliminates the forces and moments caused by the tool weight from the measured values. The concept of impedance is explained, and motivated as the basis for compliant motion control. The theoretical development leading to the simplified impedance control law is presented. The simplified impedance control law was used to provide active compliance for links 2 and 3 of the PUMA arm. The remaining four links of the PUMA were not actively used in the impedance control experiment. Force sensor accuracy was experimentally quantified and found to be sensitive to to errors in arm calibration. Execution times were determined for the major components of the impedance control algorithm. Initial testing demonstrated the ability of the impedance controller to use active compliance to reduce interface forces. Elementary compliant motion was achieved, however trajectory tracking performance requires improvements. Problem areas causing poor tracking performance are identified, and possible solutions are recommended.